Entwicklung eines Datenbankinformationssystems "Kursverwaltung" der Deutschen Unfallhilfe mit Open-Source Programmen

Fachhochschule Köln
Campus Gummersbach

Fachhochschule Dortmund

Verbundstudium Wirtschaftsinformatik

Betreuerin:
Prof. Dr. Heide Faeskorn - Woyke

vorgelegt von:
Dietmar Waschke
Jägerweg 20
44577 Castrop-Rauxel
dietmar.waschke@cityweb.de

vorgelegt am:
xx.xx.2004

Inhaltsverzeichnis

1 Abgrenzung und Gegenstand

1.1 Motivation

Das Thema Open-Source - Synonym für freie Software - wird derzeit in allen Bereichen der IT diskutiert. So hat sich das Betriebssystem Linux als populärer Vertreter von freier Software vorallem im Serverbereich etablieren können.

Dieses Betriebssystem ist bei mir seit Jahren im Einsatz und gab den Ausschlag eine Diplomarbeit im Bereich der Open-Source Programme zu schreiben. Vorallem möchte ich mit dieser Diplomarbeit anhand einer kompletten Entwicklung eines Dantenbanksystems zeigen, dass es viele leistungsfähige Open-Source Programme für die Softwareentwicklung gibt und sich der Einsatz nicht auf Server-Betriebssysteme beschränkt.

1.2 Ziel und Überblick

Diese Diplomarbeit beschreibt eine Softwareentwicklung für ein datenbankgestütztes Informationssystem. bei der Entwicklung dieses Systems wird ausschließlich frei verfügbare Software eingesetzt. Damit die Leistungsfähigkeit der eingesetzten freie Software besser beurteilt werden kann, wird in der vorliegenden Diplomarbeit beispielhaft eine Komponente einer Kursverwaltungs-Software entwickelt.

Die entwickelte Komponente ist Teil des Datenbanksystems "Kursverwaltung" der Deutschen Unfallhilfe DUH GmbH in Bochum. Diese Firma wurde ausgewählt, da dort im Rahmen einer Neukonzeption ein Datenbanksystem entstehen soll und der Einsatz freier Software favorisiert wird.

Zu Beginn der Diplomarbeit wird erläutert, was der Begriff "freie Software" bzw. "Open-Source" beinhaltet und welchen Stellenwert diese Softwareprodukte zur Zeit besitzen. Nach diesem allgemeinen Überblick werden spezielle Softwareprodukte für die genannte Softwareentwicklung vorgestellt. Anschließend werden die vorgestellten Produkte zur Entwicklung einer Komponente des Datenbanksystems "Kursverwaltung" eingesetzt. Schwerpunkt

bilden hierbei vorallem die Entwicklung der Datenbank und die Anbindung der relationalen Datenbank an die objektorientiert entwickelte Programmlogik. Dabei wird auch auf die mögliche Generierung von Softwareteilen eingegangen, welches zur Zeit einen Trend in der allgemeinen Softwareentwicklung ausmacht.

Die Diplomarbeit wird durch die Implementierung der Beispielkomponente vervollständigt und schließt mit einer Bewertung der Softwareentwicklung mit Open-Source Programmen ab.

1.3 Das Unternehmen Deutsche Unfallhilfe DUH GmbH

1.3.1 Geschäftsbereich

Das Unternehmen Deutsche Unfallhilfe DUH GmbH ist ein Schulungsbetrieb, welcher Veranstaltungen zur qualifizierten Ausbildung im Rettungswesen sowie Veranstaltungen zur Voraussetzung für die Antragsstellung der Fahrerlaubnis bei den Straßenverkehrsämtern durchführt. Die Veranstaltungen werden als Kurse bundesweit mit Hilfe von qualifizierten Ausbildern durchgeführt.

Die Zentrale und Verwaltung befindet sich in Bochum und gibt die Unternehmenspolitik für alle Standorte und Ausbildungsstätten vor. In der Zentrale werden die einzelnen Veranstaltungen geplant und Kursleiter mit der Durchführung beauftragt. Außerdem ist eine Info-Hotline als Informationsdienst zu den Kursangeboten zentral organisiert.

Die Anzahl der festangestellten Mitarbeiter in der Zentrale beträgt 12 Personen. Die Veranstaltungen werden durch ca. 250 aktive Mitarbeiter durchgeführt.[1]

1.3.2 Neuentwicklung "Kursverwaltung"

Das Unternehmen hat sich für eine Ablösung der vorhandenen Software mit einer kompletten Neuentwicklung entschieden. Grund zur Ablösung sind u.a. die mangelnde Erweiterbarkeit und unergonomische Bedienung der aktuellen Software. Das neue Datenbanksystem "Kursverwaltung" soll ausschließlich mit freien Softwareprogrammen entwickelt werden. Das zugrundeliegende Pflichtenheft wurde bereits mit dem Einsatz von freier Software erstellt.[2].

[1] Eine ausführliche Beschreibung des Unternehmens ist in der dieser Diplomarbeit zugrundeliegenden Projektarbeit zu finden.

[2] Die Beschreibung der momentan eingesetzten Software ist in der dieser Diplomarbeit zugrundeliegenden Projektarbeit zu finden. Das Pflichtenheft ist im Anhang A nochmals aufgeführt.

Bei der Neuentwicklung handelt es sich um ein datenbankgestütztes Informationssystem, welches die angebotenen Kurse der Deutschen Unfallhilfe DUH GmbH verwalten kann. Die bereits vorhandenen Daten der aktuellen Anwendung sollen in die neue "Kursverwaltung" migriert werden können.

2 Allgemeines zu Open-Source

2.1 Open-Source - freie Software

2.1.1 Begriff Open-Source

Der Begriff "Open-Source" bezeichnet ein Software-Programm, welches bestimmte Kriterien erfüllt. Die Kriterien beziehen sich überwiegend auf bestimmte Freiheiten, die das Software-Programm dem Benutzer einräumt. Damit der Begriff "Freiheit" deutlich in den Vordergrund tritt, kann man "Open-Source" ins Deutsche mit dem Begriff "freie Software" übersetzten.

Damit ein Programm als "Open-Source" bzw. "freie Software" bezeichnet werden kann, müssen dem Benutzer eingeräumt werden, dass Programm für jeden Zweck benutzen zu dürfen. Dies schließt eine Änderung sowie Verbesserungen des Programms ein. Dafür ist als zwingende Voraussetzung zu nennen, dass der Zugang zum Quellcode vorhanden ist. Ferner dürfen Kopien weiterverbreitet werden und sämtliche Verbesserungen sind der Gemeinschaft zur Verfügung zu stellen.[3].

2.1.2 Das GNU Projekt

> "Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free(1) to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed."[4]

Richard Stallmann startete mit einer E-Mail im Jahre 1983 die Entwicklung eines freien UNIX-Systems und prägte im Laufe der Jahre die Begriffe "freie Sotware" bzw. "Open-

[3] vgl. GNU-Projekt
URL: http://www.gnu.org/philosophy/free-sw.de.html
[Stand: 27. Juni 2004]

[4] Richard Stallmann, 1983
URL: http://www.gnu.org/gnu/initial-announcement.html
[Stand: 27. Juni 2004]

Source". Das GNU Projekt beschränkt sich jedoch nicht allein auf die Entwicklung von Betriebssystemen.[5].

Das GNU-Projekt listet heute eine Vielzahl von unterschiedlichen Programmen auf, die als "freie Software" zur Verfügung stehen.[6]. Damit diese und andere verfügbaren freien Programme juristisch abgesichert sind und die eingeräumte "Freiheit" nicht missbraucht wird, hat das GNU-Projekt Lizenzmodelle entwickelt. Die einzelnen Lizenzmodelle des GNU-Projekts sowie weitere Lizenzmodelle werden im Abschnitt "Lizenzmodelle" genauer erläutert.

> "GNU ist das erste ≫bewusste≪ freie Software Projekt."[7]

2.1.3 Community - Basis von Open-Source Projekten

Jedes Softwareprojekt wird durch eine kleine Anzahl von Initiatoren gestartet. Durch die Freiheiten, die "freie Software" bietet, können neben den Initiatoren von Softwareprojekten eine Vielzahl von andere Personen bei der Erstellung der Software mithelfen. Bei großen Projekten kann niemand mehr genau bestimmen, welche Personen mitgewirkt haben. Die Menge aller Mitwirkenden wird als Community bezeichnet und ist für die erfolgreiche Entwicklung von Projekten sehr wichtig.[8]

2.1.4 Lizenzmodelle

Eine Software-Lizenz ist zunächst nichts anderes als ein Vertrag, in dem eine Partei (Lizenzgeber) einer anderen (Lizenznehmer) bestimmte Nutzungsrechte an einer urheberrechtlich geschützten Software überlässt oder beschränkt. Während bei kommerzieller Software Rechte üblicherweise eingeschränkt werden, geht es bei Open Source bzw. Freier Software (mehr über die Unterscheidung unten) darum, möglichst vielen Menschen eine Veränderung des Codes zu gestatten und damit eine langfristige Verbesserung der Software zu ermöglichen.[9]

[5] vgl. GNU-Projekt
URL: http://www.gnu.org/gnu/gnu-history.html
[Stand: 27. Juni 2004]

[6] vgl. FSF/UNESCO Free Software Directory
URL: http://www.gnu.org/directory/
[Stand: 27. Juni 2004]

[7] Volker Grassmuck: Freie Software
Zwischen Privat- und Gemeineigentum. Bonn 2002, Seite 226

[8] Vgl. Volker Grassmuck: Freie Software
Zwischen Privat- und Gemeineigentum. Bonn 2002, Seite 238

[9] vgl. Open Facts
URL: http://openfacts.berlios.de/index.phtml?title=Open-Source-Lizenzen
[Stand: 27. Juni 2004]

Im Bereich der Open-Source Lizenzmodelle werden viele verschiedene Lizenzen angeboten. Sämtliche Lizenzformen werden von der Open-Source Initiative (OSI) gegenüber den Open-Source Kriterien überprüft. Die Open-Source Initiative pflegt eine Liste aller bekannten zertifizierten Open-Source Lizenzen.[10]

Nachfolgend werden die wichtigsten Open-Source Lizenzmodelle beschrieben. Um einen Eindruck zu bekommen, welches die weit verbeiteten Lizenzmodelle sind, gibt die aktuelle Statistik bei Sourceforge[11] auskunft. Die auf der Webseite zu findenden Zahlen sind nachfolgend grafisch aufbereitet.

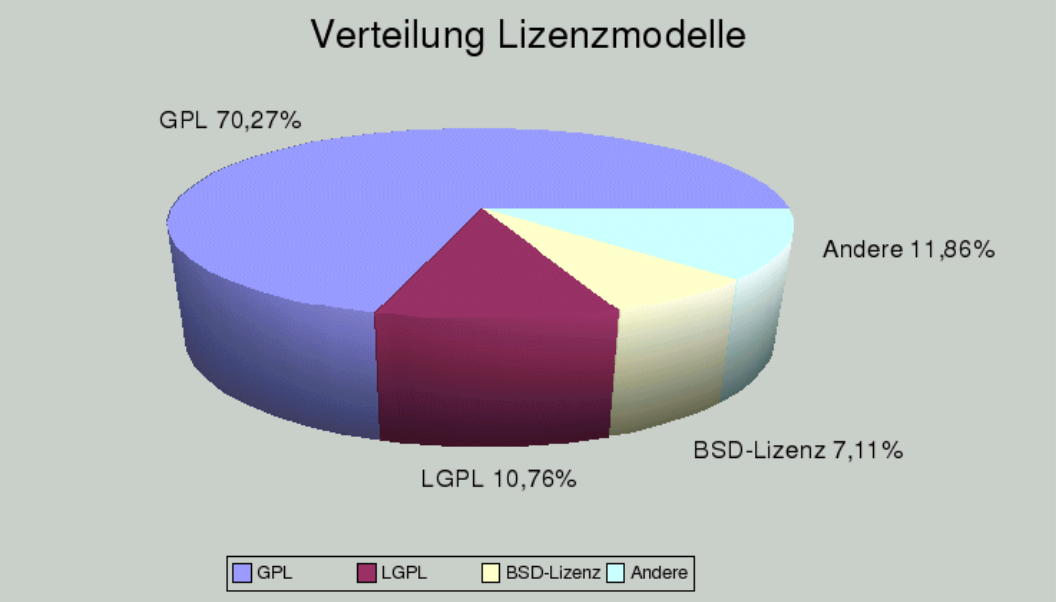


Abbildung 2.1: Lizenzverteilung, Quelle: http://www.sourceforge.net, Stand: 30.05.2004

Es ist deutlich zu erkennen, dass die GPL[12] den höchsten Anteil besitzt und damit das wichtigste Lizenzmodell im Bereich der "freien Software" darstellt.

GPL - General Public License

Die General Public License (GPL) beabsichtigt die "freie Software" abzusichern und stellt sicher, dass die Software für alle ihre Nutzer frei ist. Die Lizenzbestimmungen beinhalten

[10] vgl. Open-Source Initiative OSI
URL: http://www.opensource.org/licenses/index.html
[Stand: 27. Juni 2004]

[11] Sourceforge ist die weltweit größte Entwicklerseite im Internet
vgl. Sourceforge.net
URL: http://sourceforge.net/
[Stand: 27. Juni 2004]

[12] General Public License

die Rechte zur freien Nutzung der Software. Diese Rechte entsprechen den Eigenschaften von "freier Software". Zusätzlich bestimmt die GPL, dass abgeleitete Programme eines unter GPL stehenden Programms ebenfalls unter der GPL stehen müssen.[13]

Lesser-GPL

Die strikte Einschränkung der GPL, dass Weiterentwicklungen aus Programmen, die der GPL unterliegen ebenfalls unter der GPL stehen müssen schränkt die Entwicklung von z.B. Softwarebibliotheken stark ein. Gerade Bibliotheken sollen in anderen Programmen widerverwendet werden. Für diese Art der Lizensierung wurde die Lesser-GPL entworfen. Dabei werden die Rechte der GPL eingeräumt mit dem Unterschied, dass Programme die der Lesser-GPL unterstellt sind auch in proprietären Programmen, welche u.U. einer anderen Lizenz unterstellt sind verwendet werden können. Lediglich Änderungen an den Programmen, die der Lesser-GPL unterstellt sind müssen wieder frei zugänglich gemacht werden.[14]

BSD-Lizenz

Die BSD-Lizenz (Berkeley Software Distribution) enthält die wesentlichen Freiheiten der GPL wie die freie Verfügbarkeit des Quellcodes, die freie Weitergabe und die Möglichkeit zur Veränderung. Sie ist jedoch bei weitem nicht so einschränkend. Die Software darf, mit oder ohne Änderungen, als Quellcode aber auch als Binary verbreitet werden. Die BSD-Lizenz kann auch in kommerzielle Systeme eingebunden werden und enthält ebenfalls die Kann-Vorschrift der Open-Source-Definition. Die modifizierte Software darf also mit geschlossenem Quellcode sogar unter konventionelle Lizenzen gestellt und privatisiert werden. Ein wesentliches Merkmal der BSD-Lizenz: Sie verlangt, dass der Copyright-Vermerk bei weiterentwickelten Programmen auch den ursprünglichen Autor der Software nennt.[15]

2.1.5 Bedeutung von Open-Source

[13] vgl. GPL
URL: http://www.gnu.org/licenses/gpl.html
[Stand: 27. Juni 2004]

[14] vgl. LGPL
URL: http://www.gnu.org/copyleft/lesser.html
[Stand: 27. Juni 2004]

[15] BSD-Lizenz
URL: http://oss-broschuere.berlios.de/broschuere/broschuere-de.html#N3959
[Stand: 27. Juni 2004]

in der Softwareentwicklung

Für die Entwicklung von Software werden eine Vielzahl von Werkzeugen benötigt. Als Basistechnologien stehen viele kommerzielle Produkte zur Entwicklung von Software zur Verfügung. Dabei beschränkt sich die Auswahl der Produkte meist auf einen Hersteller, da eine Zusammenarbeit herstellerübergreifend meist nicht möglich ist.

Damit diese Abhängigkeiten von Herstellern überwunden werden können, wird auch bei der Softwareentwicklung auf "freie Software" zurückgegriffen. Für den Bereich Softwareentwicklung werden bei Sourceforge 11.450 (Stand Juli 2004) [16] Projekte aufgelistet.

Durch den bei "freier Software" verfügbaren Quellcode können die Produkte den eigenen Bedürfnissen angepasst werden. Allein die Analyse des Quelltextes zu Lernzwecken bedeutet einen unschätzbaren Wert.[17]

Als Nachteil gegenüber kommerziellen Produkten wird oft der fehlende Support genannt. In vielen Fällen stehen allerdings hinter den "Open-Source" Projekten aktive Entwicklergemeinschaften, die vorallem über das Internet Unterstützung bieten. Dabei ist die Reaktionszeit äußerst schnell und kann sich mit einem kommerziellen Support durchaus messen. Durch den engen Kontakt zur Entwicklergemeinschaft fallen Fehler schneller auf und werden auch kurzfristig beseitigt. Die Qualität durch schnelle Fehlerbeseitigung ist sehr hoch.[18]

Eine der Hauptargumente für den Einsatz "freier Software" ist der Preis, da "freie Software" überwiegend gratis eingesetzt werden kann. Allerdings kann sich dieser Vorteil bei Einsatz von "freier Software", welche nicht ausgereift ist bzw. nur eine kleine Entwicklergemeinschaft schnell als Nachteil herausstellen. In Fällen von nicht ausgereifter Software oder fehlender bzw. kleiner Entwicklergemeinschaft können hohe Folgekosten entstehen.[19]

Beim Einsatz von "freier Software" ist genau abzuwägen, ob die gebotene Leistung der Produkte ausreicht und ob die Entwicklergemeinschaft der Produkte das Prodkt in die richtige Richtung weiterentwickelt. Bei Einsatz in größeren Softwareprojekten ist die Haftungsfrage zu klären. "Freie Software" wird immer unter Ausschluß der Haftung angeboten. Dies kann bei kommerziellen Produkten u.U. individuell vereinbart werden.[20]

[16] vgl. Sourceforge
URL: hhttp://sourceforge.net/softwaremap/trove_list.php
[Stand: 27. Juni 2004]

[17] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 3

[18] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 3

[19] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 3

[20] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 4

Im nachfolgenden Kapitel werden freie Softwareprodukte zur Entwicklung eines datenbankgestützten Informationssystems vorgestellt. Die Auswahl wurde anhand der genannten Kriterien vorgenommen.

3 Softwareentwicklung mit Open-Source

3.1 Projektverwaltung

3.1.1 Konfigurationsmanagement

Das Softwareentwicklungsprojekt muss über ein Konfigurationsmanagement sicherstellen, dass die verschiedenen Versionen (Konfigurationen) der Entwicklung jederzeit hergestellt werden können. Bereits vorgenommene Auslieferungen müssen in der ausgelieferten Konfiguration wiederhergestellt werden können, so dass Fehler und die Entwicklung des Projektes nachvollziehbar bleiben.[21]

Für diese Aufgaben dient eine Versionsverwaltung. Im Open-Source Bereich zählt CVS (Concurrent Version System) zur Zeit zum Standard. Die meisten großen Open-Source Projekte und die Server von Sourceforge arbeiten mit CVS.[22].

Das Konzept von CVS basiert auf einem zentralen Repository, welches alle Dateien beinhaltet, die einer Versionskontrolle unterliegen. Auf dieses Repository können verschiedene Entwickler zugreifen. Dies ermöglicht ein Arbeiten mit vielen Entwicklern als Team.. Beim Zugriff werden die zentralen Dateien jedem Entwickler als Kopie zur Verfügung gestellt. Nachdem Veränderungen durch Entwickler vorgenommen wurden, werden die Dateien mit dem Repository abgeglichen und die Änderungen in das zentrale Repository übernommen. Das Repository speichert dabei die Änderungen so, dass vorherige Versionen jederzeit rekonstruiert werden können. Es existieren weiterhin Methoden um den Arbeitsstand des kompletten Projektes zu einem definierten Zeitpunkt zu sichern und dieses ggf. später wiederherzustellen.[23]

Durch die hohe Verbreitung und dem De-facto-Standard im Open-Source Bereich findet das CVS System auch für die Entwicklung der "Kursverwaltung" Anwendung.

[21] Vgl. Martin Backschat / Stefan Edlich: J2EE-Entwicklung mit Open-Source Tools. Heidelber/Berlin 2004, seite 212 ff

[22] vgl. Wikipedia
URL: http://de.wikipedia.org/wiki/CVS
[Stand: 27. Juni 2004]

[23] Vgl. Martin Backschat / Stefan Edlich: J2EE-Entwicklung mit Open-Source Tools. Heidelber/Berlin 2004, seite 215 ff

3.1.2 Buildunterstützung mit Ant

Damit die entwicklete Software jederzeit durch eine komplette Kompilierung zur Verfügung stehen kann, bedarf es eines Buildmanagements. Zur Unterstützung des Buildmanagements und zur Automatisierung von Aufgaben bei der Erstellung der Builds dient das Open-Source Werkzeug Ant der Apache Foundation.[24]. Als Vorläufer von Ant wird oft das Build-Tool "make" genannt, welches vorallem in der Unix-Welt eine hohe Verbreitung hat. Sämtliche Verarbeitungsschritte, zur Erstellung eines Builds werden in XML erstellt und dabei die von Ant definierte Syntax verwendet. Dabei ist Ant so flexibel, dass es durch eigene Definitionen von Aufgaben individuell erweitert werden kann.[25]

Für die Applikationsentwicklung der "Kursverwaltung" wird Ant zur Kompilierung der einzelnen Schichten des Clients eingesetzt. Außerdem werden die angebotenen Erweiterungen zur Generierung von Datenbankschematas genutzt. Dies wird im Kapitel 5 genauer beschrieben.

3.1.3 Offene Fehlerkommunikation mit Bugtracking

> "Unter einem Bug versteht man im Allgemeinen ein unerwünschtes bzw. unbeabsichtigtes Verhalten von Soft- oder Hardware, das zu Fehlern führt."[26]

Für die spätere Kommunikation mit dem Anwender und zur internen Verfolgung der Fehlerbehebung ist ein Bugtrackingsystem notwendig. Die bekannteste Implementierung ist Bugzilla, ein Bugtrackingsystem, welches ursprünglich für den Web-Browser Mozilla entwickelt wurde. Bugzilla bietet eine HTML-Schnittstelle für den Endbenutzer und eine umfangreiche Funktionalität zur Verwaltung der gefundenen Fehler an.[27]

3.2 Entwicklungszenario "Kursverwaltung"

Bei der Neuentwicklung der "Kursverwaltung" für die Deutsche Unfallhilfe DUH GmbH handelt es sich um eine kaufmännisch/administrative Software. Im Pflichtenheft für die

[24] vgl. Apache Foundation
URL: http://ant.apache.org/
[Stand: 27. Juni 2004]

[25] Vgl. Martin Backschat / Stefan Edlich: J2EE-Entwicklung mit Open-Source Tools. Heidelber/Berlin 2004, Seite 107

[26] Vgl. Martin Backschat / Stefan Edlich: J2EE-Entwicklung mit Open-Source Tools. Heidelber/Berlin 2004, Seite 235

[27] vgl. Bugzillan
URL: http://www.bugzilla.org/about.html
[Stand: 27. Juni 2004]

neue Software ist festgehalten, dass sämtliche Kursdaten verwaltet werden müssen. Dies schließt die Disposition von Kursleitern und die statistische Auswertung von Kursen ein. Der kundennahe Arbeitsplatz "Hotline" soll mit einer schnellen Auskunftsmöglichkeit einen Überblick über die Veranstaltungen bekommen können. Die neue Software soll auf eine gemeinsame Datenbasis zugreifen können (Client/Server) und eine grafische Oberfläche bieten.[28]

Für dieses Entwicklungsszenarion werden nachfolgend die Entscheigungen für den Einsatz bestimmter Open-Source Software für die Entwicklung erläutert und die passenden Open-Source Tools für die jeweiligen Bereiche vorgestellt.

3.3 Objektorientierte Softwareentwicklung

Durch den Einsatz von grafischen Oberflächen, Einführung mehrschichtiger Client/Server Architekturen, verteilte Datenhaltung und vieles mehr hat die Komplexität der Softwareentwicklung stark zugenommen.

> "Bei der objektorientierten Softwareentwicklung werden nicht nur Daten und Funktionen beschrieben, sondern auch ihr Zusammenhang sowie die Beziehungen zu ihrer Umwelt, d.h. zu anderen Daten- und Funktionseinheiten lsssen sich differenziert definieren. Diese Definitionen von Wechselbeziehungen und Abhängigkeiten sind im objektorientierten Modell immer dort vorhanden und gegenwärtig, wo sie zu berücksichtigen gilt (von der Analyse bis zur Codierung) und man ihnen die entsprechende Verantwortung zuschreibt. Dies macht die Arbeit einfacher und ermöglicht es, einen hohen Komplexitätsgrad zu bewältigen" [29]

3.3.1 Unified Modeling Language UML

> "Die Unified Modeling Language (UML) ist eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresasteme."[30]

Die UML beinhaltet verschiedene Diagrammtypen. Für die Entwicklung des Informationssystems "Kursverwaltung" kommen aufgrund der Komplexität der Anwendung die Diagrammtypen

[28] Alle Anforderderungen an die Software können der dieser Diplomarbeit zugrundeliegenden Projektarbeit entnommen werden. Das Pflichtenheft befindet sich im Anhnang A
[29] Bernd Oestereich: Objektorientierte Softwareentwicklung, Oldenburg 201, Seite 27
[30] Bernd Oestereich: Objektorientierte Softwareentwicklung, Oldenburg 201, Seite 193

- Anwendungsfalldiagramm
- Klassendiagramm

zur Anwendung.

Anwendungsfalldiagramm

Das Anwendungsfalldiagramm beschreibt die Zusammenhänge zwischen einer Menge von Anwendungsfällen und den daran beteiligten Akteuren. Das Anwendungsfaldiagramm stellt primär keinen Designansatz dar, sondern dienst als Hilfsmittel zur Anforderungsermittlung und zur Darstellung des internen Verhaltens des zukünftigen Systems.[31]

Für die Kursverwaltung wurden bei der Anforderungsanalyse die Anwendungsfalldiagramme genutzt. Ein Beispiel zeigt die folgende Abbildung.

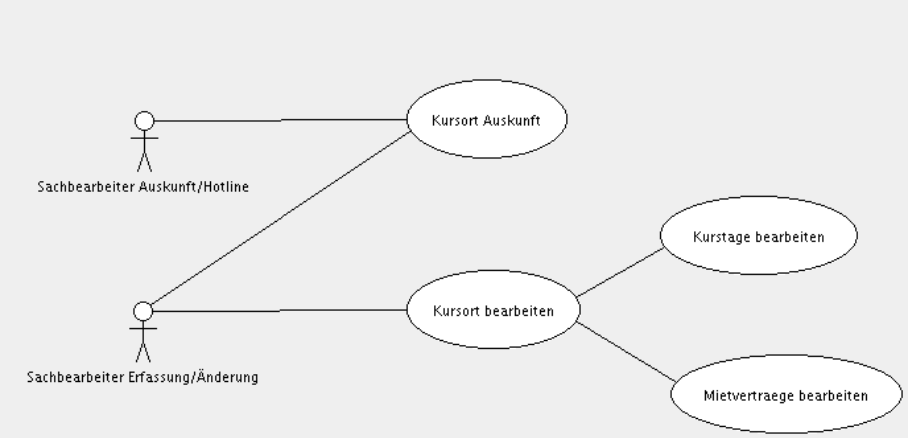


Abbildung 3.1: Beispiel Anwendungsfalldiagramm der Kursverwaltung

Klassendiagramm

"Eine Klasse definiert die Attribute und Operationen ihrer Objekte."[32]

Für die Softwareentwicklung ist ein Objekt ein individuelles Exemplar von Dingen oder Begriffe der realen Welt. Jedes Objekt besitzt einen Zustand, der durch die Attributwerte bestimmt wird. Zwischen einzelnen Objekten können Beziehungen bestehen.[33]

[31] Bernd Oestereich: Objektorientierte Softwareentwicklung, Oldenburg 2001, Seite 196
[32] Helmut Balzert: Lehrbuch der Softwaretechnik. Heidelberg Berlin 2000, Seite 154
[33] vgl. Helmut Balzert: Lehrbuch der Softwaretechnik. Heidelberg Berlin 2000, Seite 156

Ein Klassendiagramm veranschaulicht die Klassen indem die Attribute und Methoden angegeben werden. Außerdem werden im Klassendiagramm beziehungen zwischen den Klassen angegeben.

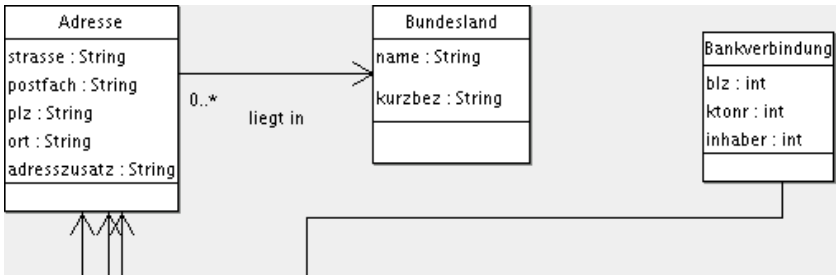


Abbildung 3.2: Ausschnitt aus OOA Klassendiagramm "Kursverwaltung"

Für die Erstellung der Anwendungsfall- und Klassendiagramme wird die Open-Source Software argoUML[34] eingesetzt. Diese Software erstellt alle Diagramme der UML Notation und dient damit der Dokumentation der Entwicklung. Außerdem besteht die Möglichkeit die UML-Modelle mit einem standarisierten XMI-Export in anderen Programmen wiederzuverwenden und dort z.B. Java-Code direkt zu generieren.

3.4 Programmiersprache

3.4.1 Objektorientierte Programmierung

Damit die Softwareentwicklung durchgehend objektorientiert durchgeführt werden kann, muss eine objektorientierte Programmiersprache zur Implementierung zur Verfügung stehen. Eine Programmiersprache gilt als Objektorientiert, wenn diese eine Abstraktion bietet. Unter Abstraktion versteht man ein abstraktes Modell, dass Objekte und deren Zustände mit anderen Objekten kommunizieren lässt, ohne dass die Implementierung der Fähigkeiten der Objekte offengelegt sein muss. Eine weitere Eigenschaft objektorientierter Sprachen ist die Kapselung. Hierunter versteht man, dass Objekte den internen Zustand anderer Objekte nicht ändern können. Ein Objekt legt selbst fest, wie über geeignete Methoden der Zustand geändert werden kann. Dies stellt die Schnittstelle zur Kommunikation mit anderen Objekten dar. Weitere Voraussetzung für objektorientierte Sprachen sind Polymorphie und Vererbung. Unter Polymorphie versteht man, dass unterschiedliche Klassen auf die gleiche Nachricht unterschiedlich reagieren. Vererbung hingegen organisiert

[34] Vgl. argoUML OJB
URL:http://argouml.tigris.org/
[Stand: 27. Juni 2004]

und erleichtert Polymorphie, indem Vererbung eine Spezialisierung von bereits vorhandenen Objekten erlaubt.[35].

3.4.2 Die Programmiersprache Java

Die Programmiersprache Java ist eine vollständig neu entworfene Sprache. Dabei ist die Syntax ähnlich wie bei C++. Im Gegensatz zu C++ ist Java eine vollständig objektorientierte Programmiersprache in der Tradition von Smalltalk[36] und integriert mit Multithreading[37] und strukturieren Excepitonhandling[38] sowie einer umfangreichen grafischen Bibliothek anspruchsvolle Neuerungen auf dem Gebiet der Programmiersprachen.[39]

> "Java-Programme sind in einem Netzwerk übertragbar, da sie in Bytecode kompiliert werden, der plattformunabhängig ist. Der Bytecode kann auf jedem Rechner eines Netzwerks laufen, der über eine virtuelle Java-Maschine (JVM) verfügt. Diese virtuelle Maschine übersetzt den Bytecode in einen Code, der von der Computerhardware verarbeitet werden kann. Dadurch spielen Unterschiede zwischen den einzelnen Computerplattformen keine Rolle mehr und es sind keine verschiedenen Programmversionen nötig."[40]

Die Programmiersprache Java wird hauptsächlich von der Firma Sun Microsystems vorangetrieben. Diese stellt Implementierung der virtuellen Maschine für verschiedene Platt-

[35] vgl. Wikipedia
URL:http://de.wikipedia.org/wiki/Objektorientierte_Programmierung
[Stand: 27. Juni 2004]

[36] Smalltalk ist eine dynamisch typisierte objektorientierte Programmiersprache und zugleich eine vollständige Entwicklungsumgebung, die in den 70er Jahren am Xerox PARC Forschungszentrum durch Alan Kay, Dan Ingalls, Adele Goldberg und andere entwickelt wurde
Wikipedia
URL:http://de.wikipedia.org/wiki/Smalltalk_%28Programmiersprache%29
[Stand: 27. Juni 2004]

[37] Multithreading wird die Fähigkeit eines Betriebssystems genannt, einzelne Tasks in scheinbar gleichzeitig ablaufende Prozesse aufzuteilen, um so z.B. deren Reaktionszeiten zu verringern oder auf Mehrprozessorsystemen zu beschleunigen.
Wikipedia
URL:http://de.wikipedia.org/wiki/Multithreading
[Stand: 27. Juni 2004]

[38] Eine Ausnahme oder Ausnahmesituation (engl. exception) bezeichnet in der Computertechnik die Möglichkeit, einen unvorhergesehenen Zustand zu entdecken und angemessen zu behandeln.
Wikipedia
URL:http://de.wikipedia.org/wiki/Exception
[Stand: 27. Juni 2004]

[39] vgl. Guido Krüger: Handbuch der Java Programmierung. München 2002, Seite 37

[40] Net-Lexikon
URL:http://www.net-lexikon.de/Java.html
[Stand: 27. Juni 2004]

formen bereit. Außerdem werden komplette Entwicklungskits angeboten. Sämtliche angebotenen Entwicklungskits sowie die dazugehörige virtuelle Maschine unterliegen der Sun Binary License.[41]

3.4.3 Einsatz von Java zur Entwicklung der "Kursverwaltung"

Obwohl die Firma Sun Microsystems nach heutigem Stand Java nicht als Open-Source freigeben möchte[42], wurde für die Entwicklung des datenbankgestützten Informationssystems "Kursverwaltung" diese Sprache gewählt, da die Sprache kostenlos verfügbar ist und für die Entwicklung eine objektorientierte Sprache mit Netzwerkunterstützung und umfangreicher grafischer Bibliothek benötigt wird. Hierzu ist eine Open-Source Alternative bislang nicht bekannt.

3.5 Entwicklungsumgebung

3.5.1 Integrierte Entwicklungsumgebung

Eine Integrierte Entwicklungsumgebung (IDE) bietet dem Entwickler neben einem Code-Editor Zugriff auf viele andere Hilfsmittel. Die Anforderungen an eine IDE sind mittlerweile sehr hoch und sollen neben den Aufgaben wie editieren und übersetzen auch Codierungsassistenz und Refactoring[43] bieten. Damit eine IDE alle Aufgaben erfüllen kann, sind diese üblicherweise erweiterbar.[44].

3.5.2 IDE Eclipse

Die Firmen Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft und Webgaid Ende des Jahres 2001 das Eclipse-Board gegründet,

[41] vgl. Sun Microsystems
URL:http://java.sun.com/j2se/1.4.2/j2sdk-1_4_2_04-license.txt
[Stand: 27. Juni 2004]

[42] vgl. Heise Online
URL:http://www.heise.de/newsticker/meldung/46034
[Stand: 27. Juni 2004]

[43] Refactoring, deutsch meist unzutreffend mit Refaktorisierung und manchmal mit Refaktorierung bezeichnet, ist ein Begriff aus der Informatik, spezieller der Softwareentwicklung. Durch Refaktorisierung verbessert man Lesbarkeit und Struktur (Architektur) eines Computerprogramms, dessen Funktionalität bleibt aber erhalten.
Wikipedia
URL:http://de.wikipedia.org/wiki/Refactoring
[Stand: 27. Juni 2004]

[44] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 9 und 10

welches sich zur Aufgabe gemacht hat eine die Eclipse IDE zur professionellen Softwareentwicklung zu fördern. Die Initiative und die erste Finanzierung erfolgte durch IBM. Mittlerweile sind viele namhafte Firmen hinzugekommen.[45]

Anfang 2004 wurde die Eclipse Foundation gegründet. Diese Foundation ist eine non-Profit Gesellschaft. Eins der Hauptziele ist die Entkopplung von IBM. Es soll anderen Softwarefirmen den Einstieg in die Unterstützung der Weiterentwicklung von Eclipse erleichtern.[46]

Eclipse läuft unter Java und bietet damit den Vorteil, dass die Entwicklungsumgebung für verschiedene Betriebssysteme verfügbar ist. Die Herausragende Eigenschaft von Eclipse ist das Plug-In Konzept. Durch dieses Konzept lässt sich Eclipse beliebig erweitern und ist damit nicht ausschließlich auf die Codierung von Java beschränkt.[47].

Weitere Eigenschaften, die Eclipse zu einer herausragenden IDE machen sind der leistungsstarke Editor, welcher umfangreiche Fehlerkennzeichnung und Refactoring des Source-Codes bietet und die Eigenschaft Entwicklungsprojekte in logische Sichten aufzuteilen. Die logischen Sichten werden in Eclipse Perspektiven genannt und werden immer für eine bestimmte Aufgabe genutzt. Lautet die Aufgabe Source-Code zu editieren, wird hierfür die entsprechende Perspektive genutzt.[48]

Eclipse ist der Common Public License (CPL) unterstellt, welche nach der Open-Source Initiative (OSI) eine zertifizierte Open-Source Lizenz darstellt.[49]

3.6 Datenbank

3.6.1 Relationale Datenbanken

Relationale Datenbanken zeichnen sich durch eine einfache Strukturierung der zu speichernden Daten in Tabellen aus. Dabei werden in Tabellen meist objekte der realen Welt

[45] vgl. Eclipse.org
URL: http://www.eclipse.org/org/index.html
[Stand: 27. Juni 2004]

[46] vgl. Entwickler.com
URL: http://entwickler.com/itr/news/show.php3?nodeid=82&id=13612
[Stand: 27. Juni 2004]

[47] Vgl. Martin Backschat / Stefan Edlich: J2EE Entwicklung mit Open Source Tools. Heidelberg 2004, Seite 42 und 43

[48] vgl. Kristian Köhler
URL: http://www.oio.de/public/warum-eclipse.htm
[Stand: 27. Juni 2004]

[49] vgl. Open-Source Initiative OSI
URL: http://www.opensource.org/licenses/cpl.php
[Stand: 27. Juni 2004]

gespeichert. Die Tabellen können dabei untereinander in Beziehung stehen. Die relationalen Datenbanken setzen konsequent das Prinzip der Datenunabhängigkeit um. Dies bedeutet, dass die Anwendungsschicht von der physischen Optimierung der Speicherunstrukturen der Datenbank getrennt ist. Eine weiterer Grund für den Erfolg relationaler Datenbanken ist die Normierung der Datenbanksprache SQL[50], welche es ermöglicht Anwendungsentwicklung unabhängig vom Datenbankhersteller zu betreiben.[51]

3.6.2 MySQL

> "Der Datenbankserver MySQL ist die populärste Open-Source-Datenbank der Welt. Mit über fünf Millionen aktiven Installationen wurde MySQL schnell zum Kernstück von vielen geschäftskritischen Hochlast-Anwendungen.."[52]

Die Datenbank MySQL wird von der schwedischen Firma MySQL AB entwickelt und vertrieben. Dieses Unternehmen zählt sich zu einem Open-Source Unternehmen der zweiten Generation. Dies bedeutet, dass das Unternehmen das Produkt MySQL einmal unter der Open-Source Lizenz GPL (General Public License) und daneben MySQL unter einer kommerzielle Lizenz vertreibt. Die kommerzielle Lizenz erlaubt dem Endanwender den Datenbankserver MySQL in proprietäre Prdukte, welche nicht unter der Open-Source Lizenz GPL stehen, einzubinden. Damit entfällt bei der kommerziellen Lizenz die Verpflichtung die proprietäre Software unter die GPL stellen zu müssen.[53]

Der Datenbankserver MySQL zählt zu den relationalen Datenbankservern. Dieser Server wird für die viele Betriebssysteme angeboten und unterstützt größtenteils den SQL99 Standard.[54]

Aufgrund der sehr hohen Verbreitung und der sehr großen Entwicklergemeinde wurde der Datenbankserver MySQL für die Entwicklung der "Kursverwaltung" ausgewählt. Die zwei Lizenzmodelle werden ebenfalls als Vorteil gewertet, da bei Erweiterung der Software je nach Bedarf ein anderes Lizenzmodell gewählt werden kann.

---

[50] Norm??? //TODO

[51] Gunter Saake / Kai-Uwe Sattler: Datenbanken &Java. Heidelberg 2003, Seite 31 ff

[52] MySQL
URL: http://www.mysql.de/
[Stand: 27. Juni 2004]

[53] MySQL
URL: http://www.mysql.de/products/mysql/index.html
[Stand: 27. Juni 2004]

[54] MySQL
URL: http://www.mysql.de/products/mysql/index.html
[Stand: 27. Juni 2004]

3.7 Anbindung relationaler Datenbanken

3.7.1 JDBC - Java Datenbank Zugriff

Die Anwendung "Kursverwaltung" wird mit der objektorientierten Programmiersprache Java entwickelt. Dabei stellt sich das Problem, wie ein Zugriff von Java auf relationale Datenbanken erfolgen kann. Damit ein Datenbankzugriff unabhängig von dem Datenbankhersteller erfolgen kann existiert als Teil der Standard-API[55] von Java die Standardschnittstelle JDBC für den Zugriff auf relationale SQL-Datenbanken. JDBC ist eine datenbankneutrale Zugriffsschnittstelle die eine direkte Nutzung von SQL Anweisungen benötigt. Daher fehlt eine höherwertige Abstraktion, welche etwa die Abbildung der Java-Klassen in entsprechende Tabellen der relationalen Datenbank unterstützt.[56]

3.7.2 Objekte in relationalen Datenbanken speichern

Die objektorientierte Sprache Java definiert im Sinne eines objektorientierten Modells verschiedene Klassen. Klassen definieren die Eigenschaften und das Verhalten der von dieser Klasse isntanziierbaren Objekte. Bei der Anbindung einer relationalen Datenbank besteht die Anforderung, dass die Eigenschaften der Objekte üblicherweise in objektorientierten Sprachen als Attribute deklariert in der Datenbank gespeichert werden können. Als einfachste Möglichkeit besteht die Abbildung einer Klasse in eine Datenbanktabelle. Dabei sind jedoch die objektorientierten Ansätze der Spezialisierung (Vererbung), Aggregationen und Assoziationen nicht berücksichtigt. Für diese Abbildungen bestehen Entwurfsmuster, welche die Abbildung sämtlicher objektorientierter Ansätze ermöglichen. Die Anbindung der relationalen Datenbank und die Abbildung der Klassen soll für den Anwendungsentwickler möglichst transparent erfolgen. Insbesondere das Puffern von Objekten, die Sicherstellung der Konsistenz der Daten und der konkurrierende Zugriff muss durch eine entsprechende Laufzeitunterstützung erreicht werden.[57]

3.7.3 Object/Relational Bridge

Die Apache Software Foundation bietet als Open-Source Community eine vielzahl von Projekten an. Ein Projekt ist die Object/Relational Bridge (OJB), welches eine transparente

[55] API ist die Abkürzung für Application Programming Interface. Ein API ist die Schnittstelle, die ein Betriebssystem oder auch ein anderes Softwaresystem anderen Programmen zur Verfügung stellt.
Vgl Wikipedia
URL: http://de.wikipedia.org/wiki/API
[Stand: 27. Juni 2004]

[56] Gunter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 55 f

[57] Gunter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 247 ff

Persistenz[58] von Java Objekten gegen relationale Datenbanken ermöglicht.[59]

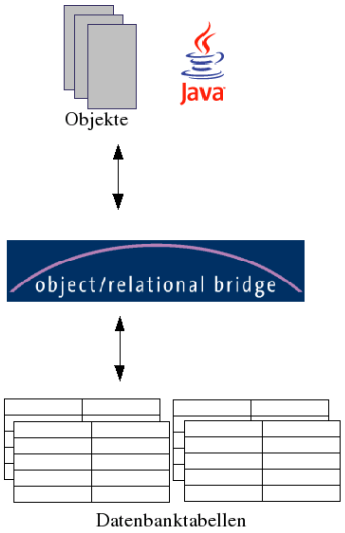


Abbildung 3.3: Überblick Object/Relational Bridge OJB

Das Projekt OJB unterstützt die Abbildung der Objekte auf die relationale Datenbank über die Beschreibung der Abbildung in entsprechenden XML-Dateien. Außerdem werden Objekt-Pufferung und verschiedene Persistenzstrategien unterstützt, so dass ein transparenter Zugriff ohne SQL auf relationale Datenbanken möglich ist. Das Projekt OJB unterstützt als Modul einige andere Open-Source Produkte, so dass der Funktionsumfang auch die Generierung des Datenbankschemas beinhaltet.

OJB stellt damit ein vollständiges Abbildungswerkzeug für die Objekt/Relatonale Abbildung zur Verfügung. Das Tool OJB wird von der Apache Foundateion als Open-Source unter der Apache License zur Verfügung gestellt.

Die Entwicklung der "Kursverwaltung" verwendet OJB als Objekt/Relationale Anbindung an die Datenbank MySQL. Dabei wird nicht nur die Unterstützung der transparenten Persistenz von Java Objekten genutzt sondern auch die möglichst vollständige Generierung des Datenbankschemas und der entsprechenden Abbildungsvorschriften aus dem Java Source Code. Die eingesetzten Techniken und die Architektur der Anwendung wird in Kapitel 5 beschrieben.

[58] //TODO
[59] vgl. Apache Foundateion
URL:http://db.apache.org/ojb/index.html
[Stand: 27. Juni 2004]

4 Architektur "Kursverwaltung"

4.1 Überblick

Die grobe Architektur der "Kursverwaltung" läßt sich auf eine zwei Schichten-Architektur zurückführen. Die zwei Schichten-Architektur bietet den Vorteil, dass die Anwendungslogik vollständig im Client implementiert wird und das Datenbankmanagementsystem die Rolle des Servers übernimmt. Auf eine drei Schichten Architektur mit Applikationsserver wurde verzichtet, da die Komplexität der zu entwickelnden Anwendung die Vorteile der drei Schichten Architektur[60] nicht ausnutzen würde. Außerdem soll die Möglichkeit bestehen, die Datenbank auch lokal an einem PC betreiben zu können. Dies ist mit einer zwei Schichten-Architektur einfacher, da am lokalen Arbeitsplatz kein Applikationsserver installiert werden muss..

Der Client wird grob in vier Teile gegliedert. Dabei kapselt der "dbAccessLayer" die Zugriffe auf das Objekt/Relationale Abbildungsprogramm OJB. Der "entitiesLayer" beinhaltet die persistenten Geschäftsobjekte. Darauf aufbauend beinhaltet der "businessLayer' die Geschäftslogik. Komplett wird die Anwendung druch den "guiLayer" der die Präsentationsschicht beinhaltet. Sämtliche Schichten bedienen sich des selbstentwickelten Frameworks und den Java Bibliotheken. Die folgende Abbildung veranschaulicht die grobe Struktur.

[60] Logging, Verteilte Architektur, hohe Skalierbarkeit und vom Applikationsserver verwaltete Persistenzschicht
Vgl. Bruce Tate/Mike Clark/Bob Lee/ Patrick Linskey: Bitter EJB.Greenwich 2003, Seite 30

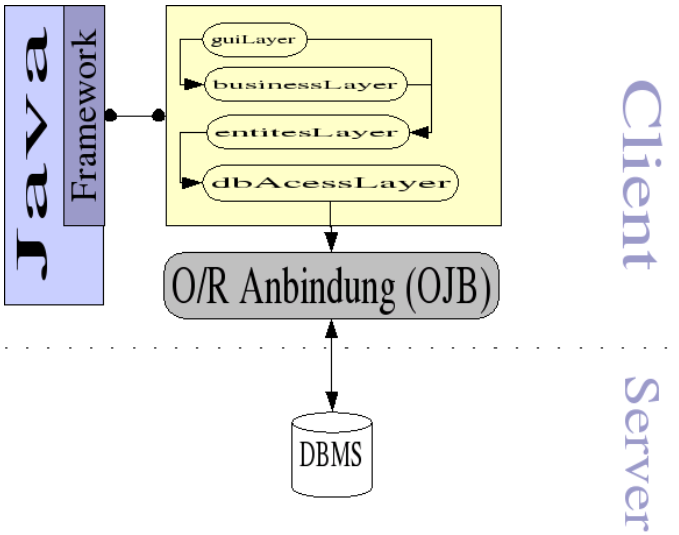


Abbildung 4.1: Architektur "Kursverwaltung"

Bei den Schichten wurde darauf geachtet, dass die oberen Schichten lediglich die unteren Schichten verwenden. Damit werden zirkuläre (zylkliche) Abhängigkeiten vermieden. Zirkuläre Abhängigkeiten erschweren die Wartbarkeit und verringern die Flexibilität.[61]

Die Schichten des Clients werden in Java als Paketnamen verwendet. Die Paketnamen werden in Java üblicherweise mit eigenem Domain-Namen benannt, allerdings sind die Namensbestandteile in umgekehrter Reihenfolge zu benennen. Die soll Namensüberschneidungen von verschiedenen Herstellern vermeiden. Für die Kursverwaltung wurde der Präfix "de.dl4dcw.duhis"[62] gewählt, so dass sich z.B. für den "dbAccessLayer" der Paketname "de.dl4dcw.duhis.dbAccessLayer" ergibt.

4.2 Datenbankzugriff

Der "dbAccessLayer" kapselt sämtliche Datenbankzugriffe. Die Datenbankzugriffe haben als Ergebnis immer ein Geschäftsobjekt oder eine Menge von Geschäftsobjekten. Die Geschäftslogik bedient sich dieser Datenbankschicht zur Ermittlung und Bearbeitung der Geschäftsobjekte. Außerdem wird für die Geschäftslogik die Möglichkeit von fachlichen Transaktionen zur Verfügung gestellt. Damit die Schnittstelle der Geschäftslogik zum Datenbankzugriff möglichst robust gegenüber Änderungen der Implementierung der Daten-

[61] Vgl .Karl Eilbrecht / Gernot Starke: Patterns kompakt. Heidelberg/Berlin 2004, Seite 8

[62] de = Deutschland
dl4dcw = wetweit eindeutiges Amateurfunkrufzeichen des Autors
duhis = Informationssystem Deutsche Unfallhilfe DUH

bankschicht bleibt und um bei entsprechenden Anforderungen auf eine drei-Schichten-Architektur mit Applikationsserver problemlos migrieren zu können, wird für den Datenbankzugriff das Design-Pattern "Fassade" eingesetzt.[63] Einzelheiten werden im Kapitel 5 erläutert.

4.3 Grafische Oberfläche

4.4 Geschäftsobjekte

4.5 Geschäftslogik

4.6 Framework

[63] Vgl. Karl Eilebrecht / Gernot Starke: Patterns kompakt. Heidelberg 2004, Seite 50f

5 Entwicklung "Kursverwaltung"

5.1 Datenhaltung

5.1.1 Geschäftsobjekte

Bei der Entwicklung der Datenbank als Basis für die persistente Speicherung der Geschäftsobjekte wurde die bereits im Einsatz befindliche Datenbank der Deutschen Unfallhilfe DUH GmbH als Grundlage genommen. Die vorhandene Datenbank befindet sich in keiner Normalform und die Daten sind recht unstrukturiert gespeichert. Die Tabelle der Kursorte hat z.B. 164 Spalten und speichert nicht nur Daten der Kursorte, sondern z.B. auch Daten des Mietvertrages.

Name	Type	Width	Decimal	Index	NULL
wegbeschr2	Character	80			
besonders	Character	80			
kuoplz	Character	5			
ansprache	Character	80			
ansprache2	Character	80			
raummiete	Character	80			
mietdauer	Character	31			
rmiete1	Numeric	7	2		

Abbildung 5.1: Ausschnitt Kursort-Tabelle der alten Datenbank

Für die objektorientierte Entwicklung wurden alle Datenbanktabellen gesichtet und einzelne Geschäftsobjekte identifiziert. Diese Geschäftsobjekte werden in der neuen Software durch Java-Klassen modelliert. Die Modellierung wird mit Hilfe des Tools argoUML vorgenommen und dient gleichzeitig als Dokumentations- und Abstimmungswerkzeug. Nach der Sichtung der alten Tabellen sind folgende Geschäftsobjekte - in der Darstellung als Klassendiagramm - identifiziert:

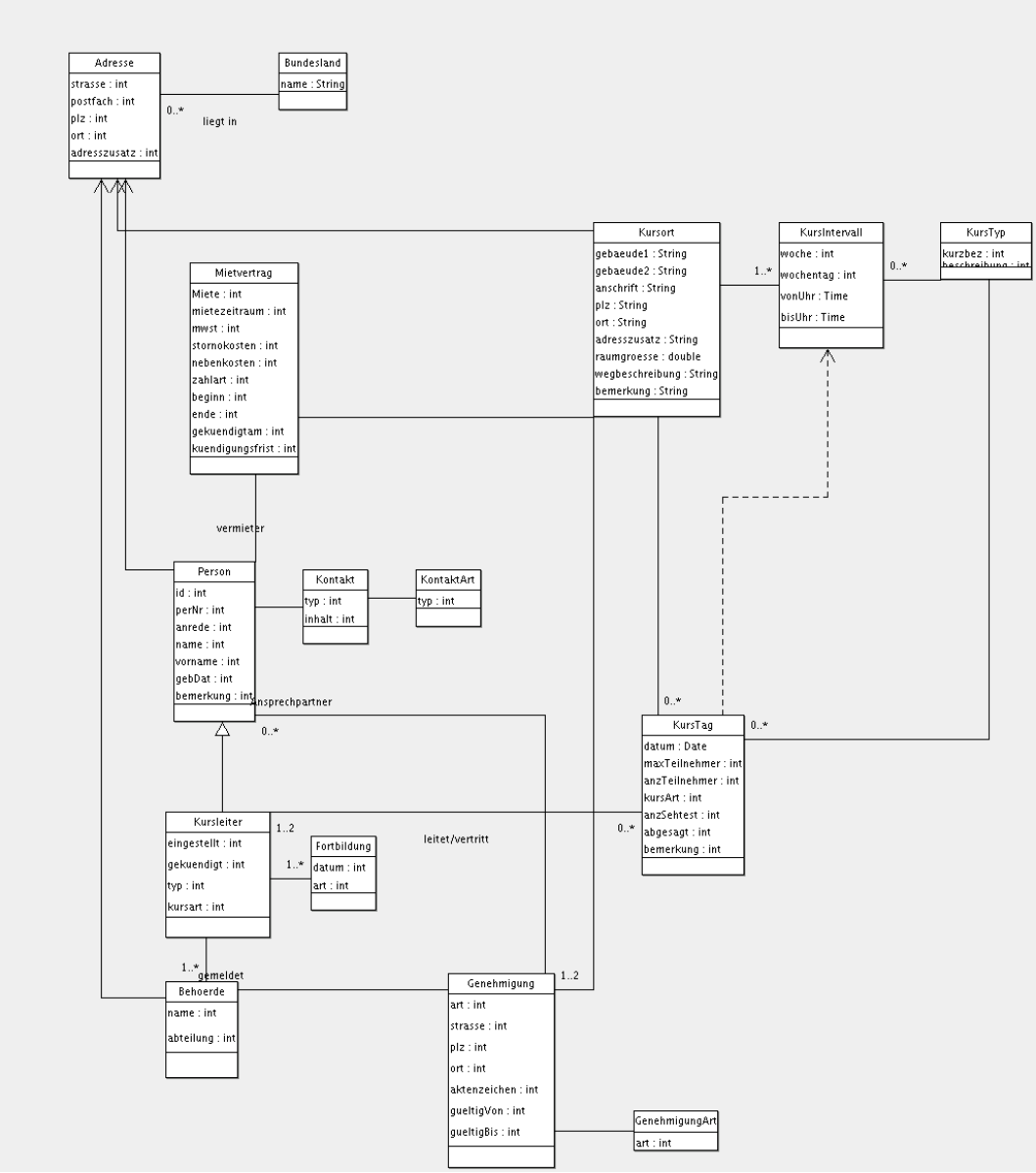


Abbildung 5.2: Geschäftsobjekte "Kursverwaltung"

Die Geschäftsobjekte sind als Java-Klassen implementiert und im "business Layer" der neuen Software "Kursverwaltung" verankert. Damit die Geschäftsobjekte möglichst unabhängig von der Präsentationslogik genutzt werden sollen, kommt ein Entwurfsmuster zum Einsatz. Entwurfsmuster sind bewährte objektorientierte Modellierungsansätze für bestimmte Problemstellungen. Dabei soll insbesondere vermieden werden, jedes Problem von Grund auf neu anzugehen.[64]

[64] Vgl. Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides: Entwurfsmuster deutsche Übersetzung von Dirk Riehle. Bonn 2001, Seite 1 ff.

Die Geschäftsobjekte sollen für die abhängigen Objekte (z.B. Präsentationsschicht) das Verhalten aufweisen, alle abhängigen Objekte bei Änderungen zu benachrichtigen.

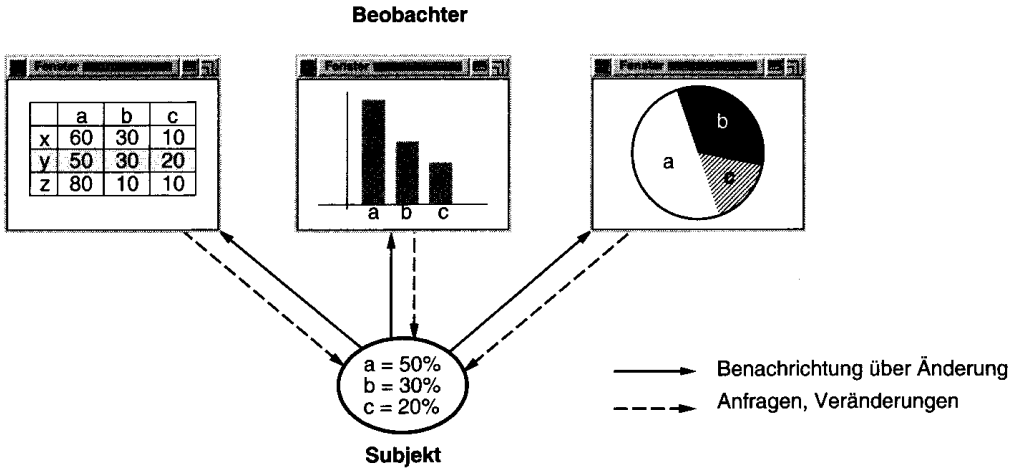


Abbildung 5.3: Beobachtermuster, Quelle: Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides: Entwurfsmuster deutsche Übersetzung von Dirk Riehle. Bonn 2001, Seite 288

Dies ermöglicht eine Präsentationsschicht, die sich aktualisieren kann, sobald sich Geschäftsobjekte ändern. Das beschriebene Verhalten wird mit dem Verhaltensmuster "Beobachter" bzw. "Observer" gelöst. Dieses Muster besitzt in der Darstellung als Klassendiagramm folgende Struktur:

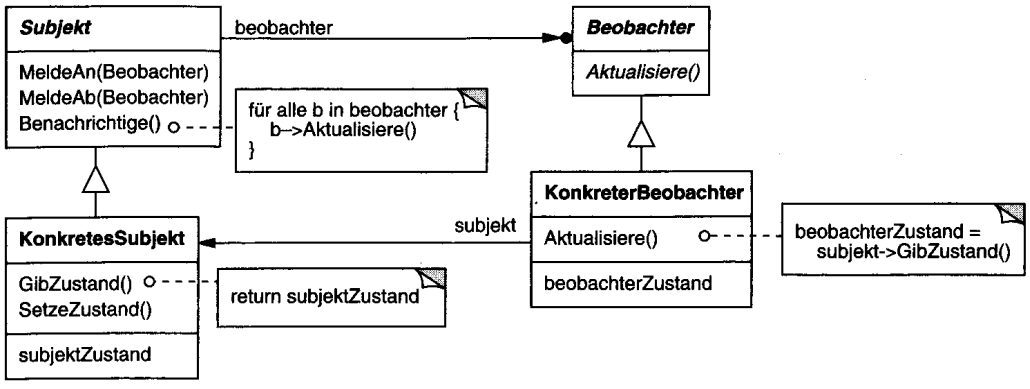


Abbildung 5.4: Struktur Beobachtermuster, Quelle: Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides: Entwurfsmuster deutsche Übersetzung von Dirk Riehle. Bonn 2001, Seite 289

Die Klassenbibliothek des eingesetzten Java Entwicklungs-Kits (Java SDK 1.4.2) beinhal-

tet zur Implementierung dieses Verhaltensmuster bereits die Klasse "Observable", welche die Funktionalitäten zum An- und Abmelden von Beobachtern und das Benachrichtigen bereitstellt. Ein Beobachter muss dann das Interface Observer implementieren und kann sich dann bei einem "Observable" registrieren.[65]

Die Geschäftsobjekte werden alle als Spezialisierung (Vererbungshierarchie) der Klasse "Observable" realisiert und erhalten damit einen Teil des Beobachter-Musters. Die Präsentationsschicht wird dann den Part des Beobachters übernehmen.

5.1.2 Datenbankschema

Das Datenbankdesign wird durch die Geschäftsobjekte bestimmt. Die Schwierigkeit besteht darin, die Klassen der Geschäftsobjekte aus dem objektorientierten Paradigma auf eine relationale Datenbank abzubilden. Die Abbildungsregeln lassen sich Gliedern in Abbildungen von

- einfache Klassen
- Klassen mit Beziehungen
- Klassen in Vererbungsstrukturen

.

Für die Abbildung der Geschäftsobjekte wird das Objekt-Relationales Abbildungsprogramm OJB von Apache eingesetzt. OJB stellt dabei die nachfolgend beschriebenen Abbildungsregeln zur Verfügung. Die Abbildungen werden zur Laufzeit so zur Verfügung gestellt, dass die Persistenz der Geschäftsobjekte transparent für den Anwender und Entwickler ist. Dies bedeutet auch den konsequenten Verzicht von SQL in den Java Objekten.

Die Abbildungsregeln werden von OJB innerhalb einer zentralen XML-Datei gespeichert und verwaltet. Für den Zugriff auf persistente Geschäftsobjekte bietet OJB verschiedene Zugriffsmöglichkeiten an, welche frei gewählt werden können. Dadurch ist eine hohe Skalierbarkeit auch für größere Anwendungen gesichert.

OJB unterstützt eine Reihe von relationalen Datenbanken. Daruter auch kommerzielle Produkte sowie das in dieser Anwendung bevorzugte Open-Source Produkt MySQL. Durch die Unterstützung einer vielzahl von Datenbankherstellern ist die Anwendung unabhängig von der Datenbank.[66]

[65] Vgl. Guido Krüger: Handbuch der Java-Programmierung. München 2002, Seite 238 f.

[66] MySQL
URL: http://db.apache.org/ojb/features.html
[Stand: 27. Juni 2004]

Nachfolgend werden die möglichen Abbildungsstrategien beschrieben und direkt der Einsatz von OJB bei den einzelnen Szenarien berücksichtigt. Die Szenarien werden anhand einzelner Beispiel Geschäftsobjekte der Anwendung "Kursverwaltung" beschrieben.

Abbildung einfacher Klassen

Die einfachste Art der Abbildung sind Klassen ohne direkte Beziehung zu anderen Klassen. Als Beispiel wird aus der Anwendung das Geschäftsobjekt "Bundesland" gewählt.

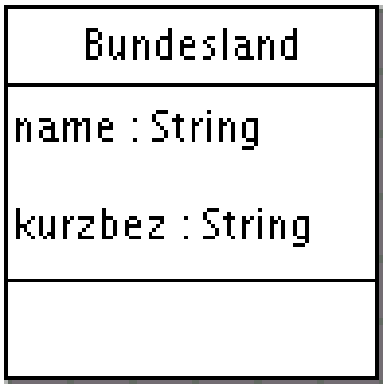


Abbildung 5.5: Geschäftsobjekt Bundesland

Das Attribut "name" speichert den Namen des Bundeslandes (z.B. Nordrhein-Westfalen) und das Attribut "kurzbez" die offizielle Abkürzung des Bundeslandes (z.B. NRW). Die Klasse "Bundesland" als Geschäftsobjekt der Anwendung implementiert diese Attribute und stellt die Attributwerte über Methoden anderen Objekten zur Verfügung.

Die Abbildung auf eine relationale Datenbank für dieses Objekt gestaltet sich einfach. Die Klasse entspricht einer Tabelle der relationalen Datenbank und die Attribute sind auf die Spalten der Tabelle abgebildet.

In relationalen Datenbanken werden zur Vermeidung von Anomalien Normalisierungen gefordert. Damit mindestens die zweite Normalform erreicht werden kann muss für jede Datenbanktabelle ein Primärschlüssel existieren.[67] Dieser eindeutige Schlüssel wird bei der Implementierung der Klasse "Bundesland" als Attribut hinzugefügt, damit aus der Klassendefinition die Datenbankabbildung vollständig ist und die zweite Normalform erreicht werden kann.

[67] Vgl. Hermann Sauer: Relationale Datenbanken. Bonn 1998, Seite 227

Für die Anwendung von OJB muss die Java Klasse des Geschäftsobjekts lediglich das Interface "Serializable" der Java-Bibliothek implementieren. Der Ausschnitt aus dem Sourcecode der Java Klasse "Bundesland" zeigt die beschriebene Implementierung:

```
public class Bundesland extends Observable implements Serializable{

    private int id;
    private String name;
    private String kurzbez;
```

Die Abbildungsregeln für die relatinale Datenbank sind für OJB in einer zentralen XML-Datei abgelegt. XML ist eine Metasprache, welche es erlaubt strukturierte Inhalte in lesbarer (nicht binärer) Form zu speichern. Die Sprache ist durch das Konsortium W3C standarisiert und bietet sich insbesondere für Dateiformate an.[68]

Von den Entwicklern von OJB ist eine Struktur für die XML-Datei vorgegeben. Dabei werden die Metadaten hierarchisch für die Datenbank, die darin enthaltenen Tabellen und die in den Tabellen enthaltenen Felder gegliedert. Für die Abbildung der beschriebenen Klasse "Bundesland" werden vorallem die Eigenschaften der Felder definiert. Dabei werden hauptsächlich die Datentypen und die jeweiligen Namen der Attribute und Tabellenspalten sowie speziell die Kennzeichnung des Primärschlüssels angegeben. Das Beispiel zeigt die beschriebenen Abbildungsregeln für das Geschäftsobjekt "Bundesland":

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE database SYSTEM "http://db.apache.org/torque/dtd/database_3_0_1.dtd">

<database name="DUHDB">
<table name="bundesland">
        <column name="id"
                javaName="id"
                type="INTEGER"
                primaryKey="true"
                required="true"
        />
        <column name="name"
                javaName="name"
```

[68] Vgl. Thomas Kobert: XML. Kaarst 1999, Seite 55 f

```
            type="VARCHAR"
            size="40"
        />
        <column name="kurzbez"
            javaName="name"
            type="VARCHAR"
            size="3"
        />

[...]
```

Aus den Abbildungsregeln ergibt sich folgende relationale Datenbanktabelle, welche mit dem dargestellten SQL-Befehl erzeugt werden kann:

```
CREATE TABLE bundesland
(
    id integer,
    name VARCHAR (40),
    kurzbez VARCHAR(3),
    PRIMARY KEY(id)
);
```

Die Implementierung der Abbildungsregeln in der XML-Datei sowie das Erzeugen der Datenbank kann durch OJB automatisiert werden. Hierzu werden die Klassen der Geschäftsobjekte durch spezielle Kommentare ergänzt. Die Erweiterung durch Kommentare stellt eine attribut-orientierte Programmierung zur Verfügung indem die Klassen durch Beschreibungen (Metadaten) in den Kommentaren ergänzt werden. Diese Kommentare werden anschließend durch das Open-Source Tool XDoclet ausgewertet und es werden Dateien aus diesen Beschreibungen maschinell generiert.[69]

OJB liefert mit dem Persistenzframework eine Version von XDoclet mit aus, welche es erlaubt die Metadaten für die Beschreibung der Abbildung in entsprechende Kommentaren zu hinterlegen. Die Kommentare folgen dabei einer festen Syntax und beginnen immer mit "@ojb.". Hinter dem Punkt folgt Klassifizierung, welche Art von Daten beschrieben werden. Im Beispiel der Klasse "Bundesland" werden die Arten "class" und "field" beschreiben, welche die Abbildung der Klasse auf die Datenbanktabelle und die Abbildung der Attribute auf die Datenbankspalten vornimmt:

[69] XDoclet
URL:http://xdoclet.sourceforge.net/xdoclet/index.html
[Stand: 27. Juni 2004]

```
**
 * @ojb.class table="bundesland"
 *            include-inherited="true"
 *            documentation="Bundeslaender der BRD"
 */
public class Bundesland extends Observable implements java.io.Serializable{

    /** @ojb.field autoincrement="ojb"
     *             primarykey="true"
    */
    private int id;
    /** @ojb.field length="40"
    */
    private String name;

    /** @ojb.field length="3"
     */
    private String kurzbez;
```

Diese Art der Beschreibung bietet den Vorteil, dass in dem Sourcecode des Geschäftsobjektes alle Daten zusammengefasst werden. Sowohl die Daten für das eigentliche Geschäftsobjekt als auch die Daten für die Abbildung auf die relationale Datenbank. Mit XDoclet werden diese Angaben nun ausgewertet und es wird aus den Angaben die oben genannte Abbildungsdatei in XML und ein Datenbankschema in einem weiteren XML-Format generiert. Das generierte Datenbankschema wird von OJB aufgegriffen und in SQL für die verwendete Datenbank transformiert. Das generierte SQL kann dann direkt das Datenbankschema erzeugen.

Die Schritte zur Generierung des Datenbankschemas und die daran beteiligten Dateien fasst die folgende Abbildung zusammen:

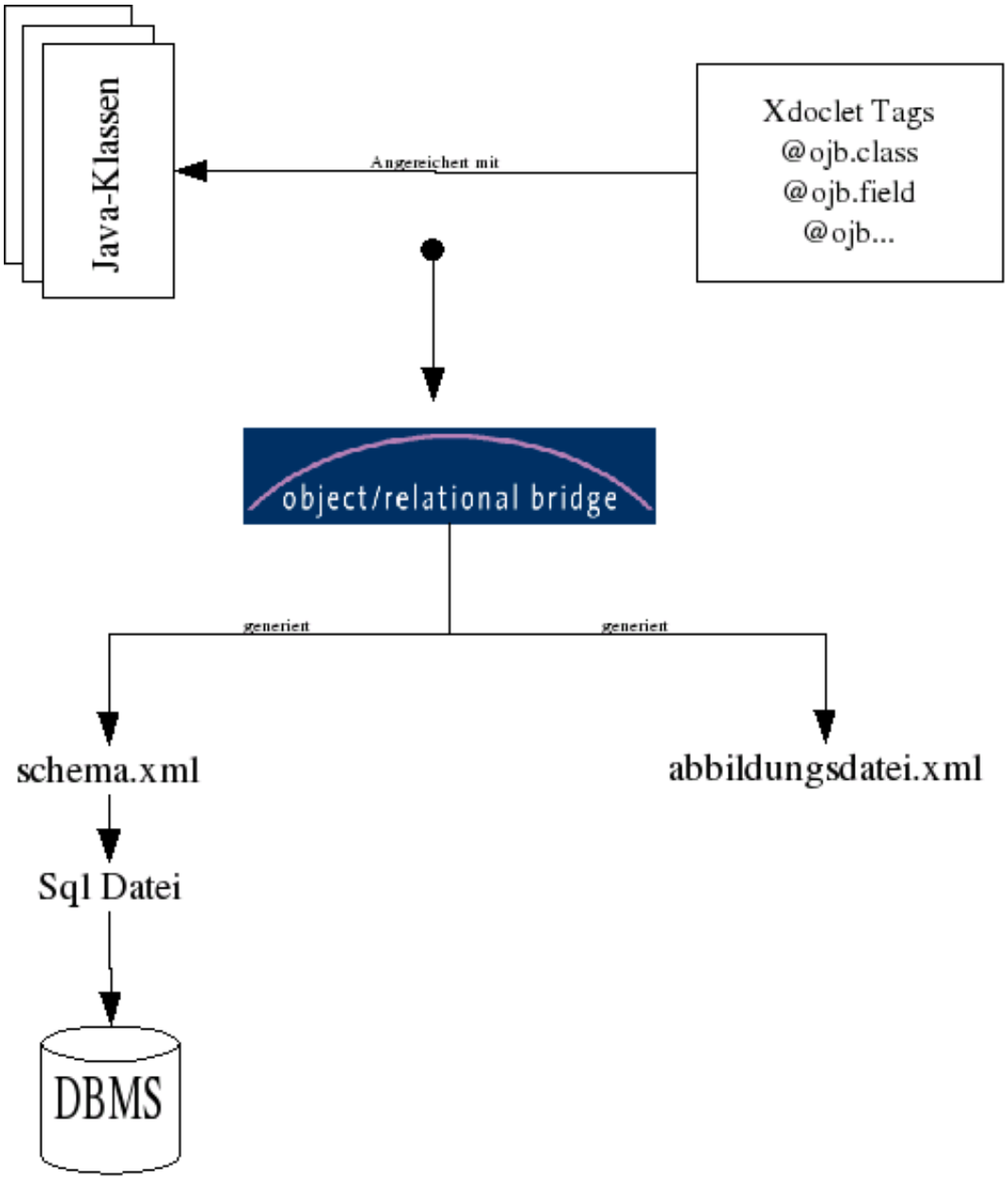


Abbildung 5.6: Generierung der Datenbank mit OJB

Die nachfolgenden Abschnitte beschreiben die komplexeren Abbildungen von Objekten mit Beziehungen und Vererbungsstrukturen. Dabei wird der Prozeß der Generierung nicht nochmal beschrieben. Es werden lediglich die Unterschiede und die Besonderheiten bei der Generierung zu den bisher beschriebenen Funktionalität aufgezeigt.

Abbildung Klassen mit Beziehungen

Klassen mit Beziehungen haben im UML-Modell eine Assoziation zu anderen Klassen. Diese Assoziationen sind meist in eine Richtung navigierbar. Die Abbildung dieses Szenarios auf eine relationale Datenbank wird anhand des folgenden Beispiels erläutert:

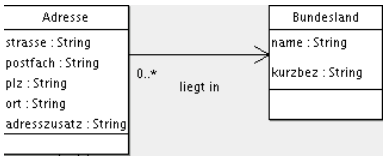


Abbildung 5.7: Beziehung Adresse Bundesland

Im Quellcode der Klasse "Adresse" wird die navigierbare Assosziation durch ein separates Attribut hergestellt. Das Attribut ist vom Datentyp "Bundesland".

```
public class Adresse extends Observable implements Serializable {

private int id;

private String strasse;

private String postfach;

private String adresszusatz;

private String plz;

private String ort;

private Bundesland bundesland;
[...]
```

Im Bereich der relationalen Datenbanken werden Beziehungen zwischen Datenbanktabellen üblicherweise über die Verwendung von Fremdschlüsseln definiert. Fremdschlüssel sind sind die Primärschlüssel der assozierten Tabelle und werden in der Tabelle eingebettet.[70] Eine Abbildung des angegebenen Beispiels in Datenbanktabellen sieht wie folgt aus:

[70] Vgl. Günter Sakke / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 250 ff.

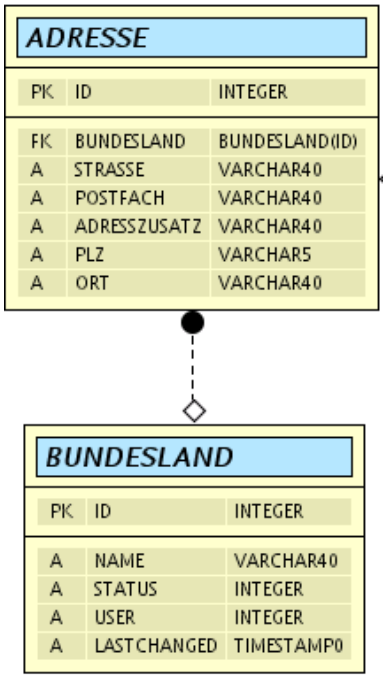


Abbildung 5.8: E/R Diagramm Beziehung Adresse Bundesland

Der Abbildung ist zu entnehmen, dass der Primärschlüssel der Datenbanktabelle "Bundesland" als Fremdschlüssel in der Datenbanktabelle "Adresse" eingebettet ist.

Diese Art der Abbildung muss durch OJB berücksichtigt werden und wird in der XML-Abbildungsdatei hinterlegt. Dafür wird zunächst das Feld Bundesland im Field-Descriptor als sogenanntes anonymes Feld deklariert. Dies bedeutet, dass in diesem Feld keine atomaren Werte der Anwendung gespeichert werden, sondern Referenzen auf andere Objekte. Damit dies nicht umständlich über die Fremdschlüsselattributwerte der Datenbank geschehen muss, sondern im Objekt direkt auf das assoziierte Objekt zugegriffen werden kann, muss das Attribut mit einem anonymen Zugriff versehen werden. Dies bedeutet, dass die Primär-/Fremdschlüsselbeziehung vor dem Entwickler versteckt wird und der Entwickler direkt mit Objektinstanzen arbeiten kannn. Damit die Referenz auf das assoszierte Objekt und die damit assozierte Datenbanktabelle aufgelöst werden können , muss diese Referenz durch einen "reference-descriptor" in der Abbildungsdatei beschrieben werden. Diese Beschreibung beinhaltet einmal die Referenz auf Ebene der Objekte und dazu die Fremdschlüsselabbildung in der Datenbank. Der Ausschnitt aus der Abbildungsdatei zeigt die Festlegungen für die Klasse "Adresse" mit einer Assoziation zur Klasse "Bundesland":

```
<field-descriptor
        name="bundesland"
        column="bundesland"
```

```
        jdbc-type="INTEGER"
        access="anonymous">
</field-descriptor>

<reference-descriptor
        name="bundesland"
        class-ref="de.dl4dcw.duhis.entitiesLayer.Bundesland">

<foreignkey field-ref="bundesland"/>

</reference-descriptor>
```

Diese Abbildungsvorschrift kann auch mit Hilfe von XDoclet (siehe //TODO) generiert werden. Für die Assoziation und deren Abbildung in der Datenbank gibt es widerum spezielle XDoclet Kommentare im Quellcode der Klasse "Adresse". Der Unterschied zur Definition von einfachen Attributen liegt darin, dass das Attribut, welches die Assoziation zum anderen Objekt hält vor der Definition der Klasse als besonders abzubildendes Attriubt deklariert werden muss. Hierzu wird ebenfalls das "@ojb.field" Kommentar verwendet. Dabei wird gleichzeitig der Fremdschlüssel beschrieben (im Beispiel ist der Spaltenname der Datenbank "bundesland" und der Datentyp "Integer" für die Datenbank vorgegeben). Die Beschreibung des Attributs erfolgt nun durch Angabe der Referenz auf die assoziierte Datenbanktabelle. Durch diese hinreichend vollständigen Beschreibungen kann durch XDoclet die Datenbank erzeugt werden und gleichzeitig die Abbildungsdatei in XML generiert werden.

```
/**
 * @ojb.class table="adresse"
 *             include-inherited="true"
 *             documentation="Alle Adressen des DUHIS"
 *
 * @ojb.field name="bundesland" column="bundesland" jdbc-type="INTEGER"
 */
public class Adresse extends Observable implements Serializable {

[...]

/** @ojb.reference class-ref="de.dl4dcw.duhis.entitiesLayer.Bundesland"
*                  foreignkey="bundesland"
*                  documentation="Fremdschluessel Bundeslaender
```

```
*/

private Bundesland bundesland;

[...]
```

Abbildung von Vererbungsstrukturen

Neben den beschriebenen Assosziationen macht eine objektorientierte Softwareentwicklung die Verwendung von Vererbungsstrukturen aus. Eine Vererbungsstruktur ist meist eine Spezialisierung von Geschäftsobjekten. Für die "Kursverwaltung" werden ebenfalls Vererbungsstrukturen genutzt. Deren Abbildung wird anhand des folgenden Beispiels beschrieben:

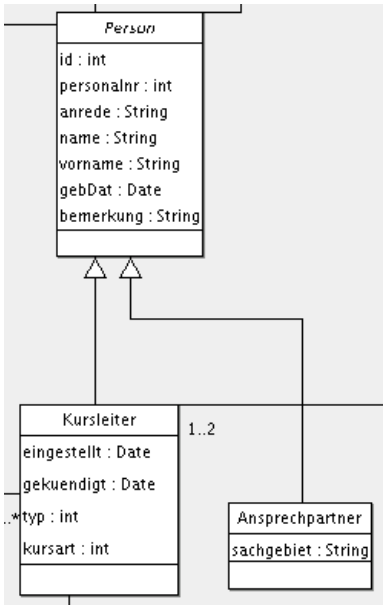


Abbildung 5.9: Beispiel Vererbungsstruktur Personen

Die Klasse "Person" ist abstrakt und speichert die geimeinsamen Daten der spezialisierten Klassen "Kursleiter" und "Ansprechpartner".

Für die Abbildung derartiger Vererbungsstrukturen existieren drei Ansätze:

- Horizontale Abbildung
- Vertikale Abbildung

- Typisierte Abbildung

Bei der horizontalen Abbildung werden die Blätter der Vererbungshierarchie als Datenbanktabelle repräsentiert. Die Tabellen erhalten dann alle Attribute der Superklasse als Datenbankspalten. Der Vorteil dieser Abbildungsmethode ist der schnelle Zugriff auf die Instanzen der Blattklassen. Dies setzt voraus, dass die Superklasse abstrakt deklariert ist oder wenige Zugriffe für die Superklasse existieren. Bei Zugriffen auf die Superklasse müssen nämlich alle Tabellen der Blätter berücksichtigt werden.

Die vertikale Abbildung vermeidet den schlechten Zugriff auf die Superklasse dadurch, dass für jede Klasse eine zugeordnete Datenbanktabelle exisitiert. Die Spalten der Tabellen richten sich dabei direkt nach den speziellen Attributen der einzelnen Klassen. Alle Tabellen werden mit Primär- und Fremdschlüsselzuordnungen miteinander verbunden. Sofern der Zugriff sich auf einzelne Klassen beschränkt, ist die vertikale Abbildung bestens geeignet. Bei Zugriffen auf alle Attriubte muss allerdings durch die verknüpften Tabellen navigiert werden, was einen Nachteil darstellt.

Es wird eine Datenbanktabelle bei der typisierten Abbildung für alle Klassen der Vererbungshierarchie angelegt. Diese Tabelle enthält alle Attribute aller Klassen in der Vererbungshierarchie als Spalten der Tabelle. Der Vorteil liegt darin, dass alle Attribute und damit auch alle Klassen schnell zur Verfügung stehen. Nachteilig wirkt sich die Speicherung von vielen Null-Werten aus, wenn eine tiefe Vererbungsstruktur vorhanden ist.

Für die Implementierung der Vererbungsstruktur wird in der Applikation "Kursverwaltung" die horizontale Abbildung gewählt. Dies bedeutet, dass für die abstrakte Klasse "Person" keine Datenbanktabelle existiert und die beiden spezialisierten Klassen "Kursleiter" und "Ansprechpartner" jeweils eine Datenbanktabelle inklusive der Spalten der Superklasse erhalten. Die Abbildungsart wurde gewählt, da sich die Zugriffe der Applikation auf die spezialisierten Klassen "Kursleiter" und "Ansprechpartner" beschränken. Zugriffe auf die abstrakte Klasse "Person" finden nicht statt.

Die horizontale Abbildung wird mittels OJB vorgenommen, indem die XML-Abbildungsdatei lediglich Angaben für die Tabellenstrukturen der spezialisierten Klassen "Kursleiter" und "Ansprechpartner" enthält:

Klassenabbildung für Ansprechpartner

```
<class-descriptor
    class="de.dl4dcw.duhis.entitiesLayer.Ansprechpartner"
    table="ansprechpartner">
```

```
    <field-descriptor
        name="id"
        column="id"
        jdbc-type="INTEGER"
        primarykey="true"
        autoincrement="true"
    >
    </field-descriptor>
    <field-descriptor
        name="personalnr"
        column="personalnr"
        jdbc-type="VARCHAR"
        length="10"
    >
    </field-descriptor>

[...]
   <field-descriptor
        name="sachgebiet"
        column="sachgebiet"
        jdbc-type="VARCHAR"
        length="100"
    >
</class-descriptor>
```

Klassenabbildung für Kursleiter

```
<class-descriptor
    class="de.dl4dcw.duhis.entitiesLayer.Kursleiter"
    table="kursleiter"
>
    <documentation>Kursleiter in der DUH Applikation</documentation>
    <field-descriptor
        name="id"
        column="id"
        jdbc-type="INTEGER"
        primarykey="true"
        autoincrement="true"
    >
```

```
    </field-descriptor>
    <field-descriptor
        name="personalnr"
        column="personalnr"
        jdbc-type="VARCHAR"
        length="10"
    >
    </field-descriptor>

[...]

    <field-descriptor
        name="eingestellt"
        column="eingestellt"
        jdbc-type="DATE"
    >
    </field-descriptor>
    <field-descriptor
        name="gekuendigt"
        column="gekuendigt"
        jdbc-type="DATE"
    >
    </field-descriptor>

[...]
</class-descriptor>
```

Diese Abbildungsarten lassen sich ebenfalls generieren, indem in den Quellcode der Klassen die XDoclet-Kommentare ergänzt werden. Hier besteht die Besonderheit, dass die abstrakte Klasse Person nicht auf die Datenbank abgebildet wird und die Attribute der abstrakten Klasse zu den spezialisierten Klassen beim Generieren der XML-Abbildungsdatei und der Datenbanktabellen hinzugefügt werden müssen.

Für die abstrakte Klasse "Person" wird das Attribut "generate-table-info = false" innerhalb der OJB-Tags verwendet. Dies bewirkt, dass die definierten Attribute nicht in der Datenbank abgebildet werden. Trotzdem werden die Attribute der Klasse exakt so ausgezeichnet als würde die Klasse abgebildet. Dies ist nötig, damit die Attribute den spezialisierten Klassen hinzugefügt werden können.

```
* @ojb.class  generate-table-info="false"
```

```
*              include-inherited="true"
*/
public abstract class  Person {

/** @ojb.field autoincrement="ojb"
 *          primarykey="true"
 */
    private int id;

    /** @ojb.field length="10" */
    private String personalnr;

    /** @ojb.field length="10" */
    private String anrede;

[..]
```

In den oberen Abschnitten wurden die Abbildungsmöglichkeiten von Klassen auf relationale Datenbanken beschrieben. Dabei sind alle Formen der Abbildungen an konkreten Beispielen der Applikation beschrieben worden. Sämtliche Geschäftsobjekte der Anwendung sind durch den mit XDoclet Kommentaren versehenen Quellcode implementiert und die Datenbank inklusive der Abbildungsdatei mit Hilfe von OJB generiert worden. Die Implementierung der Geschäftsobjekte erfolgt in einem eigenen Java Package "entitiesLayer". Das UML-Modell der Geschäftsobjekte (OOA) und das komplette E/R-Diagramm der Datenbank sind im Anhang //TODO zu finden.

5.1.3 Datenbankzugriff

Nachdem die Abbildung der Klassen auf die relationale Datenbank in der Abbildungsdatei beschrieben ist und die Datenbank entsprechend generiert wurde, werden diese Abbildungsmechanismen zur Laufzeit genutzt.

Die Zugriffe auf die Datenbank durch OJB werden in der Applikation "Kursverwaltung" in einem eigenem Paket "dbAccessLayer" gekapselt. Dieses Paket stellt Grundfunktionalitäten zur Ermittlung, Änderung und Löschung von Geschäftsobjekten bereit und wird von der Geschäftslogik genutzt. Die Kapselung erfolgt, da für den Einsatz von OJB verschiedene Abstraktionsschichten zur Verfügung stehen. Diese werden nachfolgend beschrieben und die Entscheidung zum Einsatz der entsprechenden Abstraktionsschicht begründet.

OJB gestattet verschiedene Zugriffe auf die in der relationalen Datenbank gespeicherten Geschäftsobjekte:

- JDO konforme API
- ODMG 3.0 konforme API
- OTM API
- Persistence Broker API als einfache API (Low-Level)

JDO API

Die JDO Spezifikation wird im Rahmen des Java-Community-Prozesses erarbeitet und hat das Ziel eine transparente Persistenz von Java-Objekten zu erreichen.[71]

Die Version 1.0.1 ist zur Zeit die aktuelle Version der Spezifikation[72]

Die transparente Persistenz der JDO Spezifikation zeichnet sich dadurch aus, dass Klassen per Definition persistenzfähig sein sollen. Dies wird durch einen "JDO-Enhancer" erreicht, welcher den Bytecode der Java-Klassen nach der Kompilierung verändert.[73]

Die JDO-Spezifikation wird durch viele kommerzielle Produkte untersützt (z.B. intelliBO[74]). Durch OJB wird dieser Standard ebenfalls unterstütz werden. Die aktuelle Version von OJB - zum Zeitpunkt dieser Diplomarbeit Version 1.0RC7 - unterstützt diesen Standard allerdings erst im Stadium eines Prototypen.[75]

Für die Entwicklung der Applikation "Kursverwaltung" werden keinerlei prorietäre Produkte eingesetzt und der Status der JDO Implementierung in OJB ist noch nicht stabil, so dass diese Spezifikation nicht zum Einsatz kommt.

[71] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 258
[72] Sun JDO Spezifikation
URL:http://java.sun.com/products/jdo/index.jsp
[Stand: 27. Juni 2004]
[73] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 259 f
[74] Produktbeschreibung intelliBO
URL:http://www.intellibo.de/
[Stand: 27. Juni 2004]
[75] Vgl. Apache OJB
URL:http://db.apache.org/ojb/status.html
[Stand: 27. Juni 2004]

ODMG 3.0 API

Der Standard der Object Data Management Group (ODMG) - ein Konsortium führender Datenbankhersteller - hat zur Aufgabe, die Programmiersprachenanbindung für Objektdatenbanksysteme zu spezifizieren.[76] Die aktuelle Version 3.0 definiert ein Objektmodell und spezielle Sprachen für die Objektdefinition, Objektmanipulation und Objektabfrage. Die Zielsetzung ist ebenfalls die transparente Persistenz von Objekten bzw. Klassen. Im Gegensatz zu JDO ist die Spezifikation der ODMG allerdings Programmiersprachenunabhängig und beschränkt sich deshalb nicht ausschließlich auf Java.

Die Spezifikation ODMG 3.0 stellt als zentrales Konzept - wie bei den objektorientierten Programmiersprachen - das Objekt in den Mittelpunkt. Für die Spezifizierung der Objekte ist in der ODMG 3.0 die Object Definition Language (ODL) als programmiersprachenunabhängige Sprache entwickelt worden. Mit ODL können Objekte spezifiziert werden, welche dann durch einen Präprozessor in die eigentliche Programmiersprache umgesetzt werden.[77] Die Verwendung von ODL ist nicht zwingend, so dass für die Programmiersprache Java bei der Entwicklung der Applikation "Kursverwaltung" die Geschäftsobjekte direkt mit den Konstrukten von Java implementiert werden. ODL wird vom Apache Produkt OJB nicht unterstützt. Hier wird die Implementierung in Java verlangt.

Die Manipulation von Objekten wird in der ODMG 3.0 Spezifikation durch die Object Manipulation Language (OML) spezifiziert. Diese Sprache erlaubt das Erzeugen, Löschen und Manipulieren von Objekten und deren Beziehungen.[78] Ein ODMG konformes System muss die in OML definierten Schnittstellen für die jeweilige Sprachanbindung implementieren. Die Schnittstellen umfassen die Datenbankmanipulationen (öffenen/schließen), Transaktionen und Manipulationsmöglichkeiten der Objekte. Diese Schnittstellen sind vollständig im Produkt OJB abgebildet und implementiert, so dass die Datenbankzugriffe, Transaktionen und Manipulationen ODMG 3.0 konform realisiert werden können.

Die ODMG 3.0 Spezifikation umfaßt eine Anfragesprache zur Ermittlung von Objekten. Diese Object Query Language (OQL) setzt auf die relationale Datenbanksprache SLQL-92 auf und erweitert diese um Nutzung von Objektidentitäten und Anfragen auf beliebige Kollektionen (im Gegensatz zur Mengenbeschränkug bei SQL). Ein gravierender Unterschied zu SQL ist die fehlende Unterstützung von generischen Änderungs-, Lösch- oder Einfügeoperationen bei OQL.[79] Die Abfragesprache OQL wird in OJB mit Einschränkungen[80]

[76] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 195
[77] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 200
[78] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 217
[79] Vgl. Günter Saake / Kai-Uwe Sattler: Datenbanken & Java. Heidelberg 2003, Seite 233
[80] Vgl. Apache OJB
URL:http://db.apache.org/ojb/status.html
[Stand: 27. Juni 2004]

unterstützt. Für die Anwendung "Kursverwaltung" konnten keinerlei Einschränkungen gefunden werden und die Abfragen konnten vollständig in OQL umgesetzt werden. Beispiele zur Ermittlung von Geschäftsobjekten und die daraus resultierenden Datenbankzugriffe auf Objektebene werden im Abschnitt //TODO genauer erläutert.

Der hohe Abstratkionsgrad ermöglicht eine Entwicklung der Applikation "Kursverwaltung" unter Nutzung von Geschäftsobjekten, welche allein durch die Spezifikationsangaben der ODMG genutzt werden können. Dies ermöglicht bereits jetzt einen objektorientierten Zugriff auf die persistenten Objekte, obwohl noch keine objektorientierte Datenbank im Einsatz ist. Desweiteren besteht die Möglichkeit die genutzte relationale Datenbank später durch eine objektorientierte Datenbank, welche die ODMG Spezifikation nutzt auszutauschen, ohne die Zugriffsmechanismen ändern zu müssen. Für den Zugriff auf die persistenten Objekte wird ausschließlich die OQL-Sprache genutzt. Die Einschränkungen bezüglich der Implementation von OJB sind nicht gravierend und daher vertretbar.

OTM API

Die OTM (Object Transaction Manager API) vereint die Gemeinsamkeiten der JDO und ODMG Spezifikationen. Die OTM API ist kein Standard eines Konsortiums sondern eine Implementierung der Apache Group in OJB. OTM bietet Locking-Mechanismen und die Nutzung der ODMG Abfragesprache OQL.[81]

Da sich OTM noch im Beta-Stadium befindet und die ODMG Spezifikation genutzt wird, bietet sich die Nutzung der OTM API nicht an.

Persistence Broker API

Die Persistence Broker API ist ebenfalls eine Implementierung der Apache Group und definiert keinen Standard eines Konsortiums. Diese API dient in OJB als unterste Zugriffsschicht zur Datenbank. Die Nutzung der API ist sehr einfach, da diese die Mechanismen der relationalen Datenbank nutzten und nur wenig erweitern. Es sind Abfragen in SQL möglich und die Locking-Mechanismen der Datenbank können genutzt werden. Eine Abstraktion der SQL-Sprache ist vorgesehen, so dass die meisten Abfragen durch spezielle Klassen des Persistence Brokers erstellt werden können.[82]

[81] Vgl. Apache OJB
URL:http://db.apache.org/ojb/features.html
[Stand: 27. Juni 2004]

[82] Vgl. Apache OJB
URL:http://db.apache.org/ojb/docu/pb-tutorial.html
[Stand: 27. Juni 2004]

Diese API wird für die Applikation "Kursverwaltung" unterstützend eingesetzt. Durch die geringe Abstraktion und damit starke Kopplung an die relationale Datenbank können vorallem bei Migrationsaufgaben Datenbanktabellen direkt angesprochen werden. Dabei ist sogar eine direkte Nutzung der relationalen Datenbanksprache SQL möglich.[83] Der Einsatz dieser Möglichkeiten beschränkt sich ausschließlich auf Migrationsaufgaben (siehe //TODO) und wird für die Abbildung der Geschäftsobjekte und Geschäftslogik nicht verwendet.

5.2 Geschäftslogik

5.2.1 Geschäftsobjekte

-Stichwort OQL

5.2.2 Anbindung an Datenhaltung

- Stichwort Fassade

5.2.3 Fachliche Transaktionen

- Stichwort ODMG

5.3 Grafische Oberfläche

5.3.1 Benutzer-Schnittstelle

Look and Feel

Layout

Anbindung an Geschäftsobjekte

[83] Vgl. Apache OJB
URL:http://db.apache.org/ojb/docu/faq.html#performSQL
[Stand: 27. Juni 2004]

6 Migration vorhandener Daten

7 Bewertung der Implementierung mit Open-Source

Abbildungsverzeichnis

A Pflichtenheft

A.1 Zielbestimmung

Die Firma Deutsche Unfallhilfe DUH GmbH soll mit dem Produkt DUH-IS sämtliche Daten der angebotenen Kurse verwalten und auswerten können.

A.1.1 Musskriterien

- Speicherung der Informationen zu den Kursorten, Kurstagen und Kursleitern
- Disposition von Kursleitern zu Kursen
- Statistiken zu den durchgeführten Kursen

A.2 Produkteinsatz

Eingesetzt wird das Produkt DUH-IS bei allen Mitarbeitern der Verwaltung. Die Info-Hotline benötigt lediglich Auskünfte über die vorhandenen Kursorte und Kurstage. Die Diposition und Erfassung der Daten wird von Mitarbeitern der Verwaltung vorgenommen. Die statistischen Auswertungen dienen dem Management.

A.2.1 Anwendungsbereiche

kaufmännisch/administrativer Bereich

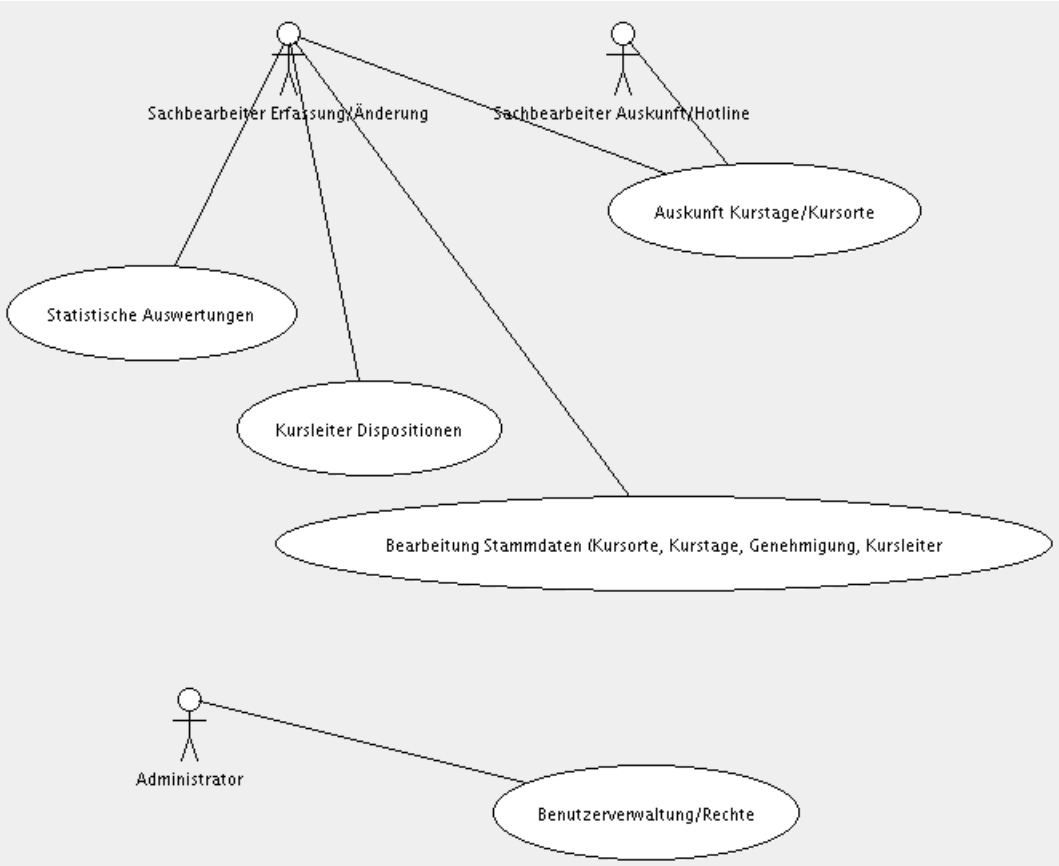
A.2.2 Zielgruppen

Die Anwender sind die Mitarbeiter der Verwaltung. Mit dem Programm sollen nicht die tatsächlichen Kursleiter arbeiten.

A.2.3 Betriebsbedingungen

Büroumgebung

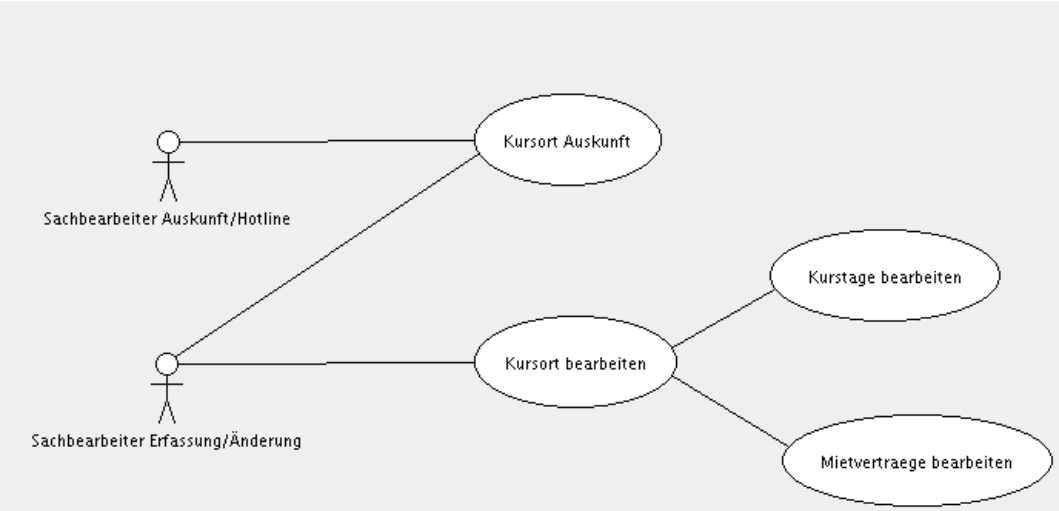
A.3 Übersichtsdiagramm



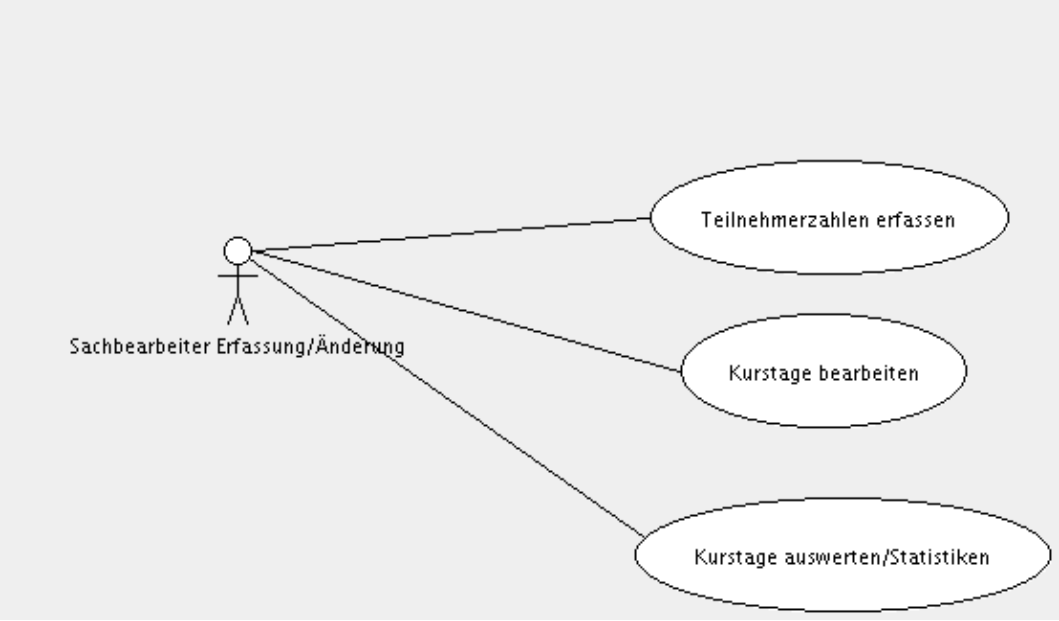
A.4 Produktfunktionen

A.4.1 Geschäftsprozesse

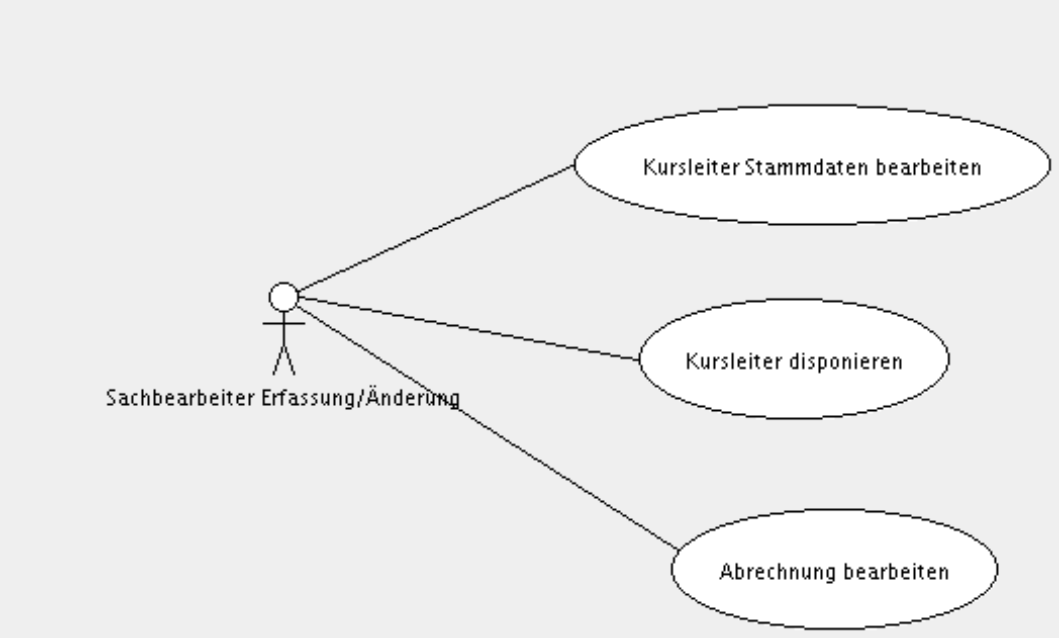
Kursort bearbeiten



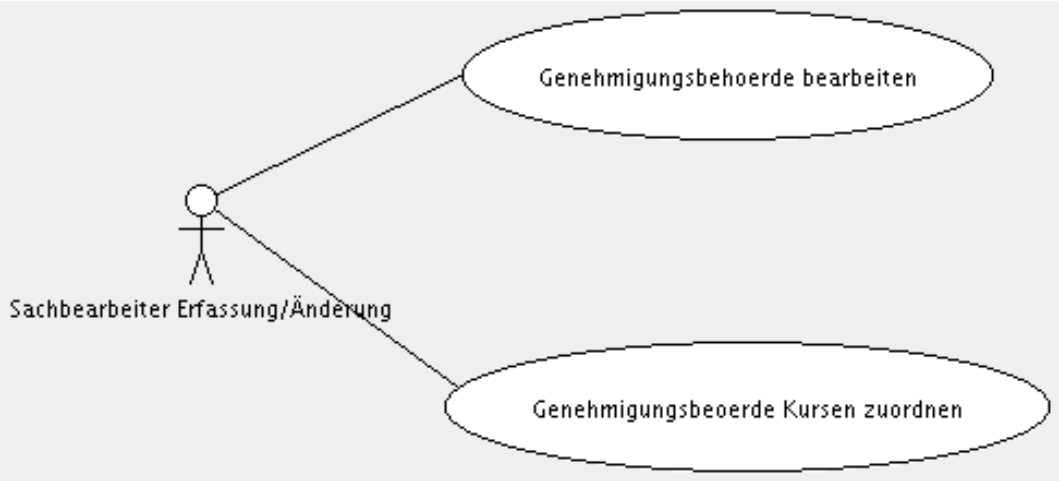
Kurstage bearbeiten



Kursleiter bearbeiten



Genehmigung bearbeiten



A.5 Produktdaten

Die anfallenden Produktdaten werden in einer Datenbank abgelegt die aus dem Open-Source Umfeld stammt. Dabei ist insbesondere das Datenbankdesign im Rahmen einer relationalen Datenbank zu erstellen.

Die zu speichernden Daten können wie folgt grob klassifiziert werden:

A.5.1 Kursverwaltung

- Kursorte (ca. 180)
- Kurse (ca. 20000)
- Genehmigungen (ca. 180)
- Mietverträge (ca. 180)

A.5.2 Disposition Kursleiter

- Kursleiter (ca. 300)
- Abrechnungen (ca. 20000)

A.6 Benutzungsoberfläche

Für die Benutzungsoberfläche soll eine grafische Benutzeroberfläche entwickelt werden. Da die Anwendung üblicherweise von Verwaltungsangestellten durchgeführt wird und lediglich für das Betriebssystem Windows Kenntnisse vorhanden sind, wird eine Benutzeroberfläche in Anlehnung an Windows angestrebt.

Es gibt mehrere Rollen mit unterschiedlichen Rechten in der Anwendung:

Sachbearbeiter Info-Hotline: lesender Zugriff auf die Stammdaten der Kurse und Kursorte
Sachbearbeiter Disponent: lesender und schreibender Zugriff auf alle Daten
Sachbearbeiter Administrator: einrichten von Benutzern und volle Zugriffsrechte

A.7 Nichtfunktionale Anforderungen

Die Anwendung muss so zu bedienen sein, dass unterschiedliche Informationen gleichzeitig aufgerufen werden können. Dies ist über Aufrufen von mehreren Fenstern in der Anwendung sicherzustellen.

Die Anwendung wird in der Zukunft im Netzwerk eingesetzt. Daher ist sicherzustellen, dass die Anwendung von Anfang an netzwerkfähig entwickelt wird.

Als Lizenzmodell wird die Open-Source Lizenz GPL gewählt.

A.8 Technische Produktumgebung

A.8.1 Software

Betriebssystem: alle Windows Versionen ab Windows 98
Datenbank: Open-Source Datenbank mit einer kostenfreien Lizenz im kommerziellen Umfeld

A.8.2 Hardware

Standard-PC

A.9 Spezielle Anforderungen Entwicklungsumgebung

Die Entwicklung wird vollständig mit Open-Source Produkten durchgeführt. Hierzu sind entsprechend zu evaluieren:

- Entwicklungsumgebung
- UML-Programm für objektorientierte Analyse und Design
- Datenbank
- weitere den Entwicklungsprozess unterstützende Programme (z.B. Dokumentation, Erstellung Datenbankschema)

B Eingesetzte Programme

Programmname	Version	Einsatzgebiet	Lizenz
ant	1.5.3	Buildmanagement und Automatisirung	Apache Licence
argoUML	0.14	Dokumentation und UML Diagramme	GPL

//TODO vervollständigen!

C Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig ohne fremde Hilfe gefertigt und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt und alle Zitate kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Castrop-Rauxel, xx.xx.2004